1996-03

User interface and database design for software database of the Computer Aided Prototyping System (CAPS)

Hong, Ruey-Wen

Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/32164

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

USER INTERFACE AND DATABASE DESIGN FOR SOFTWARE DATABASE OF THE COMPUTER AIDED PROTOTYPING SYSTEM(CAPS)

by

Ruey-Wen Hong

March 1996

Thesis Co-Advisors: Man-Tak Shing
Luqi

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE March 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE User Interface and Database Design for Software Database of the Computer Aided Prototyping System(CAPS)	5. FUNDING NUMBERS ARO #MA30989 The National Science Foundation CCR-9058453
6. AUTHOR(S) Hong, Ruey-Wen	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School, Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES) Army Research Office, 4100 S Musimi Blvd, Research Triangle Park, NC. National Science Foundation, 1800 G St, Washington, DC	10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

CAPS (Computer-Aided Prototyping System) is an integrated set of software tools that generate prototypes directly from real-time requirements. The success of CAPS depends on being able to generate the prototype quickly so that it can be evaluated, leading to the construction of a program built on true requirements thereby resulting in a better software product. The key to developing prototypes quickly is having a significant software base to choose reusable components from. The problem with the current version of CAPS is that there exists no software base storage facility. This thesis utilizes the Ontos database to build the object oriented conceptual design for the data object repository and uses TAE to create the graphical user interface to access the repository. It further explores various searching techniques to determine the best possible implementation of the repository search engine. The results of this thesis are a conceptual design that can be used to implement the software base and an interface which provides a fluid, intuitive, interactive environment in which the user will be able to manipulate the database when actually built. It further identifies the multi-level filtering technique as the best candidate for searching the database, because of its high recall, high precision, and reduced search time.

14. SUBJECT TERMS Software Reuse, Software Component, Software Database, Retrieval, Searching Process			15. NUMBER OF PAGES 320
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

USER INTERFACE AND DATABASE DESIGN FOR SOFTWARE DATABASE OF THE COMPUTER AIDED PROTOTYPING SYSTEM(CAPS)

Ruey-Wen Hong
Lieutenant Cmdr, TAIWAN ROC, Navy
B.S., Chung-Cheng Institute of Technology, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

March 1996

Author: [redacted]
Ruey-Wen Hong

Approved by: [redacted]
Man-Tak Shing, Thesis Co-Advisor

[redacted]
Luqi, Thesis Co-Advisor

[redacted]
Ted Lewis, Chairman,
Department of Computer Science

ABSTRACT

CAPS (Computer-Aided Prototyping System) is an integrated set of software tools that generate prototypes directly from real-time requirements. The success of CAPS depends on being able to generate the prototype quickly so that it can be evaluated, leading to the construction of a program built on true requirements thereby resulting in a better software product. The key to developing prototypes quickly is having a significant software base to choose reusable components from. The problem with the current version of CAPS is that there exists no software base storage facility.

This thesis utilizes the Ontos database to build the object oriented conceptual design for the data object repository and uses TAE to create the graphical user interface to access the repository. It further explores various searching techniques to determine the best possible implementation of the repository search engine.

The results of this thesis are a conceptual design that can be used to implement the software base and an interface which provides a fluid, intuitive, interactive environment in which the user will be able to manipulate the database when actually built. It further identifies the multi-level filtering technique as the best candidate for searching the database, because of its high recall, high precision, and reduced search time.

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Thanks Buddha.

Thanks for everything.

Thanks my parents.

Thanks Dr. Shing, Dr. Luqi and Dr. Berzins.

Thanks Scott, Tuan, Doan and Osman.

Thanks Mr. Albert Wong and all the CAPS team members.

Especially,

Thanks my wife Miin-Hwa and my son Eric.

I. INTRODUCTION

For the past several decades, software engineers have constantly struggled with the issue of whether or not to rewrite their applications in the latest and greatest language on the market. While newer languages offer great benefits, they always come with a price. Every year massive man-power efforts and billions of dollars of investment are expended on the task of rewriting applications. Developing an application with reuse capability becomes an important issue.

This thesis concentrates on designing an efficient software database and building a flexible graphical user interface for the software base of the Computer Aided Prototyping Systems(CAPS), which is an ongoing software engineering project in the Computer Science Department of the Naval Postgraduate School.

Chapter II describes the background knowledge and related work. Chapter III describes the current retrieval techniques(multi-level filtering retrieval process) of CAPS. Chapter IV discusses the conceptual design of data storage. Chapter V discusses graphical user interface design issues and the implementation details. Chapter VI concludes with a summary of the work done thus far and discusses future concerns. It identifies several areas which need further improvement/ research.

A. SOFTWARE REUSE

1. Introduction to Software Reuse

Software reuse is defined as the process of implementing or updating software systems using existing software assets. Software reuse is the application of reusable software assets to more than one software system. It may occur within a software system, across similar software systems, or in widely different software systems.

A reusable software asset is a software product that is cataloged in a reuse library. This includes, for example, models of distinct functional areas (i.e., domains), domain architectures requirements, designs, code, databases, database schemas, documentation, user manuals, and test suites[Ref. 1].

2. The Importance of Software Reuse

Software reuse provides a basis for dramatic improvements in quality and reliability, speed of delivery, and in long-term decreased costs for software development and maintenance. Industry has demonstrated that software reuse generates significant return on investment(ROI) by reducing cost, time, and effort while increasing quality, productivity, and maintainability of software systems throughout the software life-cycle.

A significant challenge to both management and technical staff is to improve the quality and reliability of an organization's systems in an era of decreasing budgets. The challenge is intensified by the ever increasing complexity and functionality of modern information systems. Software engineers have stressed the importance of reducing the cost, time, and effort required to build and maintain software systems. This applies to both in-house and contractor-developed systems.

Many software development costs have outstripped hardware costs and are continuing to grow. The major factors contributing to this growth of software costs are the continuing increase in the size and complexity of software systems and an international climate that calls for rapid adaptation to new situations. Software reuse offers significant potential to hold down future costs by taking advantage of previously developed software and by developing software designed for future reuse.

One response to this situation that has proven successful is the integration of software reuse principles into the software engineering process. Reusable software requires carefully analyzed and structured design that withstands thorough testing for functionality, reliability, and modularity. Accordingly, improvements manifest themselves in increased quality and reliability and in long-term decreased costs for software development and maintenance.

Reuse reduces the risks and costs of software acquisition, development, and maintenance. Other benefits of adopting reuse include improved interoperability and support for rapid prototyping activities, such as CAPS, discussed later. A well run reuse program decreases initial risk for development, maintenance, and acquisition activities by taking advantage of software already proven to be functional and reliable through prior usage.

Reuse is already an integral part of every engineering discipline. For example, mechanical engineers do not design a combustion engine from scratch for each car rolled off an assembly line; software engineers do not recreate the interaction between an application and its screen icon for each new product; and aerospace engineers do not build a solid rocket booster from ground zero for each space shuttle. In all of these examples, the architecture and design of an item is reused to produce and manage a "product line". This is also the reason why the architecture model of software reuse is so important.

Reusable software can be acquired, developed, maintained, and managed via a "product-line", approach. For example, during the past decade we have to build systems quickly and cheaply. One strategy for doing this is to move to product-line techniques like commercial firms use to achieve economies of scale.

The product-line approach will increase the software acquisition, development, and maintenance community's responsiveness to project's plan and requirements for the future. The anticipated result is a capability to deliver more responsive, higher quality products (systems) to the end user more quickly, at reduced cost, and with less risk because common software is reused.

3. The Benefits of Reuse

The biggest benefit of reuse is that it lowers the cost of systems. Software reuse will yield significant benefits to the software acquisition, development, and maintenance[Ref. 1]. Commercial firms and government organizations have experienced that software reuse will:

- Improve the quality and reliability of software-intensive systems;
- Provide earlier identification and improved management of software technical risk;
- Shorten system development and maintenance time;
- Get products to market sooner;
- Increase productivity; and
- Reduce cost.

4. Reuse Techniques and Approaches

To provide a framework for reuse activities, [Ref. 2] separates reuse into compositional and generative techniques. An additional technique, denoted as adaptation, follows from these two and rounds out the framework. Compositional techniques use building blocks and group them together. Utilizing components that are passive in nature, its emphasis is on component libraries and on organization and composition principles. Generative techniques reuse patterns of knowledge. Utilizing components that are more active in nature, it is typified by language-based generators, or by application generators which combine some reusable knowledge (patterns) with application-specific constraints to generate products (e.g., code). Adaptive techniques combine the nature of the two techniques above, but focus on transforming or modifying of descriptive (formal) specifications/languages into executable structures.

These techniques require incorporation into a comprehensive and complementing set of mechanisms or process models to be more readily put into action. Noteworthy approaches to reuse are found in [Ref. 3, 4]. Three common process models, adaptive reuse, parameterized reuse and engineered reuse, are described in [Ref. 5] that incorporate existing and newly emerging reuse practices. These could be viewed as delineating the lines of reuse formality.

Adaptive reuse is accomplished by salvaging or scavenging components from previous projects as a starting point. Occasionally, adaptive reuse involves reusing old components to create a modified version of an old system or porting them to a new platform or environment.

Parameterized reuse occurs when a standard generic base program is established as a common chassis. This base program is then used to build several different systems by extensions which meet the unique requirements of each system.

Engineered reuse is the most formal of the three processes. It involves undertaking a domain analysis to derive a domain model and software architecture. Engineered reuse applies the results of the domain analysis to domain engineering, resulting in building and controlling reusable components for use in new applications[Ref. 6].

The terms describing reuse do not stop here - the following may be of interest. Below is a list of terms extant in the reuse community that describe techniques manifesting various degrees of formality. Though some terms describe similar approaches, each seems to have an implicit reason for prescribing reuse.

- Adaptive reuse: Making use of what is currently available.
- Ad-hoc reuse: Reuse practiced even when there are no defined methods being utilized.
- Carry-over reuse: Using one version of software component 'as is' in a subsequent version within the same system.
- Architecture-centric reuse: Developmenting of components and applications based upon a generic architecture.
- Black box reuse: Composing systems by plugging together reusable components.
- Compositional reuse: Constructing new software products by assembling reusable workproducts
- Derived reuse: Software reuse accomplished via the object-oriented principle of subclassing.
- Domain specific reuse: Constructing applications within a certain domain using products, processes, and assets that are applicable to that domain.

- External reuse: Using workproducts produced in an non-organic, but related project and taken from an external library.
- Generation-based reuse: Using application generators to build new applications from high level descriptions.
- Institutionalized reuse: Reuse that is significantly practiced throughout the entire organization employing formal and planned processes.
- Internal reuse: Avoiding redundant implementation of functionality within a single project by careful design and inspection at early stages such that selected components are identified for distinct uses within a project.
- Horizontal reuse: Described earlier as reuse germane to a horizontal domain.
- Large-scale reuse: Reapplication of high-level components (e.g., requirements, architectures, designs).
- Leveraged reuse: Starting with existing workproducts in the development process and modifying as needed to meet specific system requirements.
- Library-assisted (Repository-based) reuse: The existence of one or more, possibly interconnected libraries, supports development within an application domain.
- Maintenance reuse: Similar to carry-over reuse, but stresses the use of products from an earlier version to base any enhancements to the system rather than allowing unique development from similar requirements.
- Opportunistic reuse: Relying upon software developers to identify reuse possibilities, locate components (usually created without reuse in mind), and to integrate them.
- Process-driven reuse: Practicing reuse as an integral and transparent part of both the software engineering process and broader acquisition process.
- Small-scale reuse: Reapplication of lower-level code components (e.g., subroutines, object libraries, or Ada packages.
- Systematic reuse: The planned reuse of workproducts with a well-defined process and lifecycles, with commitments for funding, staffing and incentives for production and use of reusable workproducts.
- Vertical reuse: Described earlier as reuse germane to a vertical domain

B. COMPUTER AIDED PROTOTYPING SYSTEM(CAPS)

The Computer-Aided Prototyping System (CAPS) [Ref. 7] is an on-going software engineering project at Naval Postgraduate School for developing prototypes of real-time systems. It is useful for requirements analysis, feasibility studies, and the design of large embedded systems. It also creates a software base library to support software reusability. CAPS is based on the Prototype System Description Language (PSDL) [Ref. 8], which provides facilities for modeling timing and control constraints within a software system. CAPS is a development environment, implemented in the form of an integrated collection of tools, linked together by a user-interface. CAPS provides the following kinds of support to the prototype designer:

1) timing feasibility checking via the scheduler, syntax direct editor.

2) consistency checking and some automated assistance for project planning, scheduling, designer task assignment, and project completion date estimation via the Evolution Control System,

3) design completion via the editors, and

4) computer-aided software reuse via the software base.

A CAPS prototype is initially built as an augmented data flow diagram and a corresponding PSDL program. The CAPS data flow diagram and PSDL program are augmented with timing and control constraint information. This timing and control constraint information is used to model the functional and real-time aspects of the prototype. The CAPS environment provides all of the necessary tools for engineers to quickly develop, analyze and refine real-time software systems. The general structure of CAPS is shown in Figure 1.

As Figure 1 indicates, CAPS is a collection of tools, integrated by a user-interface. The CAPS User-Interface provides access to all of the CAPS tools and facilitates communication between tools when necessary. The tools in the Figure 1 are grouped into four sections: Editors, Execution Support, Project Control and Software Base.

This thesis concentrates on the software base. Being able to retrieve the correct components when necessary is a very complex task. Much work has been, and currently is being expended on the retrieval of software components. Because this problem has not yet been solved, we explore the reuse component retrieval techniques in the following two chapters. The next chapter will focus on those techniques which have been around for some time and have been explored under CAPS. Chapter III will go further with a less explored, potentially more powerful technique.

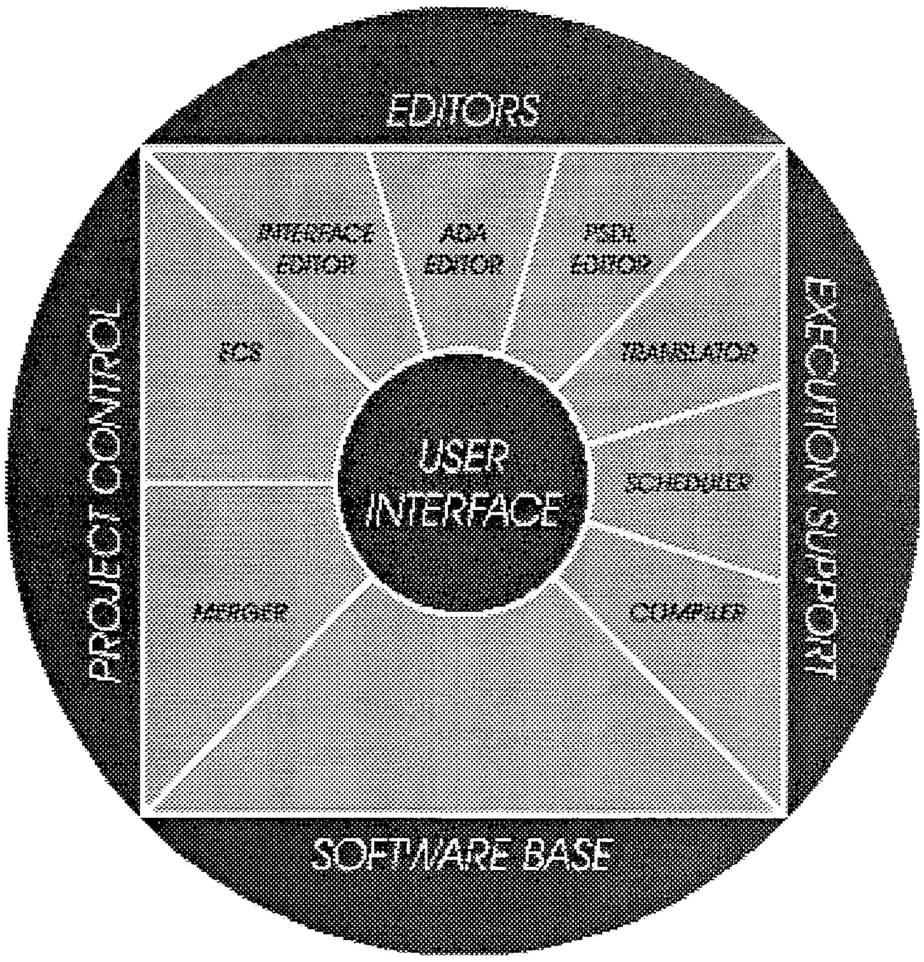


Figure 1. The CAPS Development Environment

II. BACKGROUND AND RELATED WORK

A. RETRIEVAL TECHNIQUES

Much research has been done in the area of reusable software component retrieval techniques, all of which have tried to increase the search accuracy and reduce the retrieval time. There are three major concepts related to the reusable component retrieval techniques; *representation, search*, and *measures of performance.*

Representation is the description of how the component storage is structured and facilitates the retrieval performance. For example, the multi-attribute technique, covered later, may be used for the future retrieval of a certain component. The schema of the component repository should include, at a minimum, several attributes related to the search module in order for the retrieval mechanism to do an efficient search. To make a long story short, the method of representation and the method of search work together to form a cohesive environment for information retrieval. The more refined and precise the method of representation, the easier the search mechanism becomes[Ref. 9].

A lot of research has been conducted for search mechanics too, especially in artificial intelligence and database management systems[Ref. 9]. Various search methods will be described in the following section.

Precision and recall are the most important two measures of performance[Ref. 10]. *Precision* is defined as the ratio between the number of relevant components retrieved and the number of the total components retrieved. It is the precise percentage of the useful retrieved components compared to the total retrieved. *Recall* is defined as the ratio between the number of relevant components retrieved and the number of relevant components in the library. It answers the question, "What percentage of the relevant components in the database did my query find?". Both precision and recall reach the ideal situation when the set of components retrieved is exactly the same as the set of components that are relevant.

B. METHODS FOR RETRIEVING REUSABLE COMPONENTS

The method for retrieving reusable components could be categorized by different approaches. There are three major approaches for retrieval: *Interactive Browser, Informal Specification-Based method*, and *Formal Specification-Based method.*

1. Interactive Browser

An Interactive browser is a tool which helps the user look through the software base according to its hierarchical structure. The hierarchy might be structured by keywords(ex: list, ring, stack), specific features, categorizations(ex: data structure, interface, communication), collection name, author name, etc. The purpose of the browser is to let the user manually search for the desired component by going through the various levels of the library hierarchy. It is somewhat like a World-Wide-Web (WWW)[1] browser(such as Netscape) as it has similar pros and cons.

The pros of a browser is that the user can completely control the library and view the whole library structure by browsing the various level topics. This is useful if the user is familiar with the software library and when the user is not familiar with it, it serves as an excellent tool to educate them. When the user is familiar with the software library, this technique proves to be much faster in finding relevant candidates than by other search methods.

The cons are actually weaknesses in the topics already mentioned(the pros). First, since it is only efficient when the user is familiar with the software library, it will be a problem when the library is large. For example, it is not easy to search a topic in the WWW by browsing all of the Internet sites instead of using a search engine. The only exception is when the user already knows or has an idea about the location of that topic. Even when the Internet user knows of some particular location of the desired stuff, how could s/he be sure that there is no other relevant stuff somewhere else. The same thing can happen when browsing the software base for a particular component that is needed, raising the disadvantage of the browser; to cause very low recall in retrievals since the browser has much difficulty in covering all the relevant components.

Thirdly, the browser is a totally manual controlled system, which means it is haltless unless the user terminates the process. Not only do recall and precision measurements suffer with this technique, it can be very time intensive when used on an unfamiliar, large repository.

2. Informal Specification-Based Method

The Informal Specification-Based Method includes keyword search, multi-attribute queries, and natural language searches[Ref. 11]. The difference between this method and browsing is that it involves an automatic search algorithm. In other words, the retrieval tool will halt and come out with all of the candidates related to the query. The reason we call it an informal specification

1. WWW: World Wide Web of the Internet.

is because the attributes used are more informal and English-like, which will realize only general matching capabilities. Informal specifications are most useful for domain-specific repositories, because in such contexts it is possible for the administrator of the repository to anticipate the purposes for which a component will be used, and to design a keyword structure to match the application. However, these approaches are not well suited for exploratory design of new applications, which is the main goal of rapid prototyping. Retrievals based on informal specifications or keywords are also not very selective: in addition to the relevant components, these techniques may find many irrelevant ones, thus relying on manual effort of the user for a substantial part of the work in doing the searching[Ref. 11].

a. Keyword Search

Keyword searching is based on a list of keywords, which are either known system controlled vocabularies or uncontrolled(unconstrained) vocabularies. In the case of unconstrained keywords, synonym tables are often used to find more standard words on which to perform the query[Ref. 10]. This technique is used by CAPS and the Operation Support System OSS, discussed later.

The first advantage of keyword search is that it is easy to implement due to the simple concept. Secondly, it reduces the search time for a large library, because unlike the browser, it utilizes an automatic searching algorithm. Thirdly, the user does not have to know the library structure. Whenever the user looks up the keyword list table, which is just an interface for the search engine, s/he does not have to deal with the data storage; the algorithm does it for them. It indirectly isolates the implementation of the retrieval engine and the data repository. This aspect is also enhanced by the formal specification methodology which will be discussed in Chapter III.

Measurement problems are the major downfall of keyword search. The fewer number of attributes used, the higher the recall and the lower the precision. Contrarily, the more attributes used, the lower the recall and the higher the precision. This leaves the user with a dilemma: how many keywords to specify.

b. Multi-attribute Search

Multi-attribute search is an extension of the keyword search. It is based on faceted classification. Facets, proposed by Prieto-Diaz[Ref. 12], are groups of related terms in a subject area. For example, a facet to describe the functions performed by components might be terms chosen from *find, compare, sort, update, send, receive,* etc. A scheme is developed in reference [Ref. 12] to describe Unix components using four facets: the function performed by the component, the objects that are manipulated, the data structure used, and the system to which the function belongs. This provides a better description of Unix components than a pure keyword approach. However, it still relies on an informal description, using a limited set of facets and terms. The Multi-attribute approach is used by CAPS, DRACO, RAPID, OSS, the Reusable Software Library(RSL), and the Common Ada Missile packages project(CAMP).

c. Natural Language Interface

The Natural Language Interface is used by Reusable Software Library(RSL) and is a growing field of computer science research. Users can drive the retrieval query command with their natural speaking language. It is good for the user yet it increases the difficulty of implementation due to the broad semantics of the English language.

d. AI approach

AI-based work includes [Ref. 13, 14] and some recent work by Henninger[Ref. 15], which uses a knowledge-base and statistical information to retrieve reusable components, based on keyword search from texts describing the components. However, because the characterization of the component behavior is completely informal, the behavior is unpredictable.

3. Formal Specification-Based Method

According to Webster's Dictionary, the word "formal" means definite, orderly, and methodical; it does not necessarily entail logic or proofs or correctness. Everything that computers do is formal in the sense that syntactic structures are manipulated according to definite rules. Formal methods construct the software architecture essentially with syntactic and semantic specification rules[Ref. 16].

The formal specification-based retrieval method is based on the formal method of software architecture. Recent work using semantics for software component retrieval is reported in reference[Ref. 17, 18]. The primary aim is to check that retrieved components yield the behavior specified in the user's query, therefore increasing the precision of retrieval. Using formal specifications as search keys has two main problems. The first problem is practicality: not all users are sophisticated enough to write formal specifications, much less correct ones. The second problem is that semantic matching is very time consuming, because some form of theorem proving must be done.

The Venari[2] project at Carnegie Mellon University, headed by Prof. Jeannette Wing, is devoted to retrieving components from software libraries, and has produced a number of interesting publications. Here we will not discuss their work on transactions and other infrastructural support for retrieval, but only their work on the search process.

Rollins and Wing [Ref. 19] discuss signature matching for retrieving higher-order functions from a MetaLanguage (ML) library,[3] using λProlog for matching user queries to component signatures. λProlog is used to implement matching modulo various theories, in order to support partial matches.[4] They also use λProlog to check simple pre- and post- conditions for ML functions. Although this paper demonstrates that higher-order logic is useful for such applications, we feel that higher-order logic is more powerful and expressive than necessary, and that higher-order logic tools like λProlog are too inefficient. Of course, a higher-order language like ML requires the use of higher-order types, but these are first-order expressions, so that first-order matching could be used. Rollins and Wing point out that equational reasoning could dramatically increase precision, and they also discuss the possibility of specification matching.

Zaremski and Wing[Ref. 20] extend this work. First, they consider signatures in two different senses, as the rank of a function, and as the interface of a module; the second sense involves search and retrieval of modules, not just of functions. Second, they consider a wider variety of matching procedures and their combinations, although some of these are needed only because of the awkwardness of the higher-order encoding of operation ranks (e.g., uncurrying). Third, they implemented their matching procedures in ML, experimented with retrieving functions from actual ML libraries, and presented some interesting statistics on these experiments.

In more recent work, Zaremski and Wing [Ref. 21] focus on specification matching, using the Larch/ML interface language to express pre- and post-conditions in first order logic, and the Larch prover to verify that candidate components satisfy these conditions. Various senses of matching are defined, but neither ranking nor partial semantic matching are considered. This

2. This name is from the Latin verb "to hunt".
3. For these authors, the word "signature" refers to the rank of a higher-order function, rather than to the syntactic specification of a software module, as in the algebraic tradition.
4. The Venari project uses the term "partial match" in a more restricted sense than we do; their term corresponding to our "partial match" is "relaxed match".

approach has not resulted in a practical automated method for specification based retrieval.

Because these specification-based methods are based on specification formalisms and axioms, they are free from ambiguity and can be transformed into normal representations by using logic and term rewriting rules without changing their meaning.

The disadvantage is that specifications may be difficult for designers to write. Another disadvantage is that processing times for the search algorithms may be excessive depending on the approach taken. Finally, matching formal specifications is a hard problem.

C. CURRENT RETRIEVAL SYSTEMS AND TOOLS

1. ASSET

ASSET, an acronym for Asset Source for Software Engineering Technology, is sponsored by the Advanced Research Products Agency(ARPA) organized under the STARS program. ARPA tasked Loral Federal Systems (formerly IBM Federal Systems) and its principal subcontractor, SAIC (Science Applications International Corporation) to establish the ASSET Reuse Library to serve as a national resource for the advancement of software reuse across the Department of Defense (DoD).

ASSET's mission is to provide a distributed support system for software reuse within the DoD and to help foster a software reuse industry within the United States. ASSET's initial and current focus is on software development tools, reusable components, and documents on software development methods. ASSET is participating in interoperation with other reuse libraries, such as CARDS(Comprehensive Approach for Reusable Defense Software), AdaIC (Ada Information Clearinghouse), DSRS (Defense Software Repository System), and ELSA (Electronic Library Services & Applications Lobby).

ASSET's goals are to create a focal point for software reuse information exchange, to advance the technology of software reuse and to provide an electronic marketplace for reusable software products to the evolving national software reuse industry. To achieve these goals, ASSET operates the Worldwide Software Resource Discovery(WSRD) Library, and Newsgroup services.

The ASSET staff are constantly working to catalog and add new reusable software components to the library. Consequently, to determine the latest contents of the library at any time, the ASSET account holder should log on and search the new assets listings.

ASSET is populating the Worldwide Software Resource Discovery(WSRD) Library with quality reusable software components, which can be distributed to its subscribers. The library specializes in software lifecycle artifacts, and in documents written specifically to promote software reuse and development. ASSET also interoperates with the CARDS and DSRS reuse libraries thus allowing ASSET users access to a larger variety of components.

ASSET currently uses the STARS Reuse Library (SRL) mechanism as its library interface. The primary purpose of the SRL is to serve as a library mechanism for the operation of a software reuse library. It provides a search mechanism, using a combination of faceted and attribute-value classification schemes, for finding candidate assets from the reuse library's collection. Once a list of assets has been obtained, the SRL permits the user to browse through the descriptive material about each asset as well as through the source files of the asset. After browsing the user may then select assets for extraction from the library. A copy of the files of each asset being extracted are then made available to the user for immediate downloading or for copying to their own local directory area.

ASSET has implemented its initial operating concept of library interoperation. Using TCP/IP protocol, ASSET shares software artifacts with the CARDS (Comprehensive Approach for Reusable Defense Software), AdaIC (Ada Information Clearinghouse), DSRS(Defense Software Repository System), and ELSA(Electronic Library Services & Applications Lobby) repositories, allowing users to extract components from any of the above libraries. These remote library extractions occur transparently to the user. ASSET is continuing efforts to broaden its base of cooperating libraries.

2. DRACO

The Draco project[Ref. 22] is an approach to software engineering that has had a large impact on software reusability in general. Draco approach focuses on domain engineering of software.

The most important aspect of Draco is the domain language. Software components are organized into problem areas or domains and a domain language describes objects and operations of a particular domain. There is a reusable component associated with each domain language object or operation. Since there is a potentially large number of components within a domain, a classification scheme is developed for the components called faceted classification to aid in organizing and retrieving the components.

Using faceted classification, Draco approach utilizes a multi-attribute query method. Queries are constructed by the formulation of a tuple of attributes that best characterizes a particular domain. A query session begins with the most specific query, that is, all attributes filled in. If the results of the query are unsatisfactory, the user may generalize the query by inserting wildcards for attribute values.

The advantages of faceted classification are that it is conceptually simple for users and relatively easy to implement. Because of this, the concept has been borrowed to implement the retrieval mechanisms in both RAPID (see Chapter II.C.3) and OSS (see Chapter II.C.5).

One disadvantage of multi-attribute search is that semantically similar components may not be found when their attribute definitions are different. Draco alleviates this problem by maintaining a measure of conceptual closeness for the term lists of each attribute. This way, an unsuccessful search can be tried again using an alternative but similar term in one of the attributes.

The disadvantages of faceted classification are that it is not suitable for unconstrained domains and semantically similar components may be missed from other domains, even with a conceptual closeness measure.

3. RAPID

The RAPID (Reusable Ada Packages for Information System Development) project is an ongoing effort in the Department of Defense. The objective of RAPID is to provide software engineers with quick access to reusable Ada packages in the information systems domain. The system performs reusable component classification, storage and retrieval.

RAPID uses a faceted classification scheme to organize and retrieve components and thus uses multi-attribute searches[Ref. 23]. The system is currently being beta tested but no measures of performance or quality assessments are available yet.

4. The Reusable Software Library(RSL)

The Reusable Software Library is a system designed to make software reuse an integral part of the software development process[Ref. 24]. The system couples a passive software database with interactive software design tools to help software developers find and evaluate components to meet their requirements.

Components are stored in the database with attribute values that provide a basis for search. There are two methods available to search for components, standard multi-attribute search and natural language. The multi-attribute approach provides a menu driven interface in which the user selects the attributes with which to perform the search. Alternatively, the user may express his query in the form of natural language, such as “I need a stack package.”. The system parses the input, extracts keywords from it and uses those words as attributes to perform the search.

The designers of the system report that the natural language front end is considerably easier to use but the search is significantly slower, by a factor of five to ten because of the natural language parsing overhead involved.

5. Operation Support System

The Operation Support System (OSS) is an ongoing project aimed at developing an integrated software engineering environment undertaken by Naval Ocean Systems Center. One goal of the project is to establish a Navy software library of reusable software components.

The current prototype library subsystem allows component retrieval using faceted classification, keywords, or a textual browser. The components currently stored in the library are large command, control, and communications subsystems. Since the library is in its early stages, no information is available on its performance characteristics.

6. CAPS

The CAPS project at the Naval Postgraduate School, headed by Professors Luqi, Berzins and Shing, supports rapid prototyping for hard real time embedded systems. CAPS consists of an integrated set of software tools that help design, translate and execute prototypes. These tools include an execution support system, a syntax directed/graphical editor, an evolution control system, a change merge facility, and facilities to support retrieving reusable components from a software base. The execution support system includes a scheduler and a translator for automatic generators for schedule and control code.

PSDL is the Prototyping Description Language of CAPS[Ref. 8]; it is used to specify both prototypes and production software. PSDL programs have two kinds of objects, corresponding to abstract data types and abstract state machines; they localize the information for analyzing, executing and reusing independent objects. Executable Ada modules can be associated with atomic PSDL objects, and CAPS can automatically generate “glue" code that composes these modules into a system having the structure described by PSDL. This generated code includes a schedule to enforce all real time constraints that have been declared. The system can then be compiled, executed, and tested. Error messages are produced during execution if constraints are violated. Figure 2 illustrates a prototyping life-cycle. It shows two places where component search can be used in such a life-cycle: in constructing a prototype system and in constructing a production system.

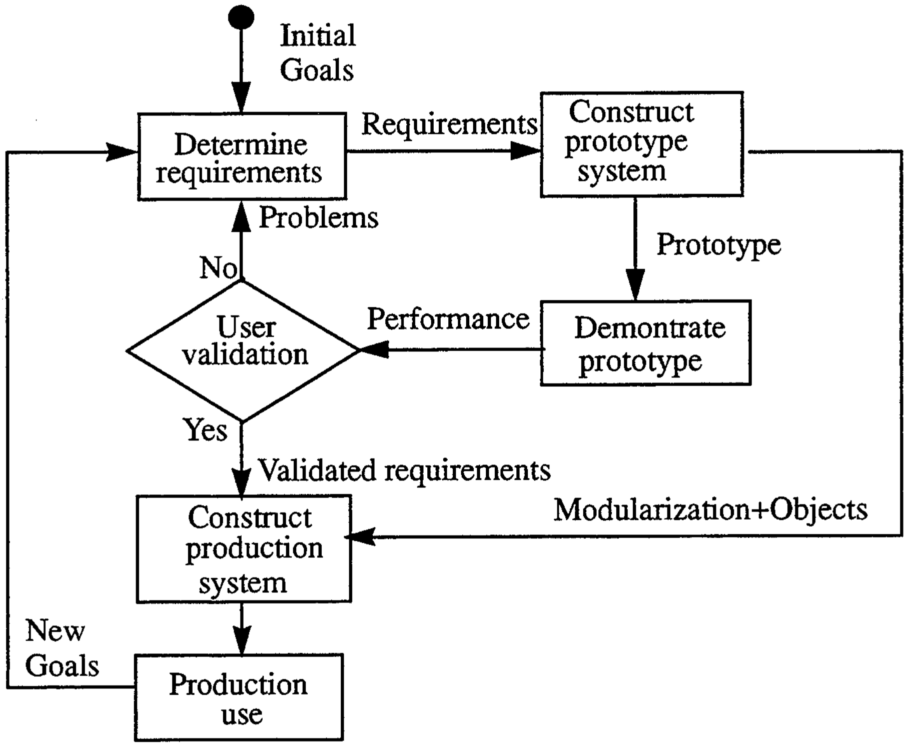


Figure 2. A Prototyping Lifecycle

Some work has done in the CAPS project on retrieving software components. Dr. Luqi suggested the use of specifications in retrieving software components[Ref. 17]. This suggestion was refined in later work, including [Ref. 25] and Steigerwald's Ph.D. thesis [Ref. 26]. In [Ref. 26] it is assumed that each component has a fully expanded[5] algebraic specification written in OBJ3 [Ref. 27, 28, 29], and that the user's query is also a fully expanded algebraic specification that the desired component should satisfy. A Prolog program was written using symbolic representations of signatures to find syntactic matches between the signature of the query and the signatures of components. For each match found, a semantic validation was done by evaluating patterns that represent the functions in the signature, first in the query specification,[6] and then (after translation) in the specifications of the matched components; the results of these

5. This means that the results of any module expressions inside a module are substituted into the module.
6. These patterns are terms that involve those functions, plus some variables, constants, and constructors, such that all other functional expressions are instances.

two evaluations are compared to determine the quality of each match. The approach developed in this series of papers is the inspiration for the approach taken in the present dissertation.

The system described in [Ref. 26] has certain technical limitations. Its semantic basis is not well developed. Also, evaluating patterns with variables gives limited information about the semantic satisfaction of a syntactic match. In addition, since patterns can involve variables that may or may not be eliminated by rewriting, depending on syntactic peculiarities of the equations, it seems possible to have semantically equivalent specifications for which pattern evaluation would give conflicting answers, so that the match in question will appear not to satisfy its semantic requirements even though it really does. In addition, the approach is limited to total syntactic matches and to unparameterized components.

Ozdemir's master's thesis [Ref. 9] describes a component retrieval system for the CAPS software base that uses keyword search and a browser. Both of these use PSDL for queries and for components. Ozdemir also provides a graphical user interface and facilities for integrating retrieved components into prototype systems, including techniques for transforming retrieved modules. A better developed version of these ideas appears in the master’s thesis of Dolgoff [Ref. 30]. This work is including retrieval of generic modules and handling of subsort matching.

III. MULTI-LEVEL FILTERING RETRIEVAL

Multi-level filtering retrieval technique(MLFRT) is a new software component searching process developed for the CAPS software base[Ref. 37]. It is the combination of interactive browsing, informal specification-based and formal specification-based methods. Figure 3 describes the Multi-level filtering architecture; the top line indicates user modification of the query in light of the final filtering results.

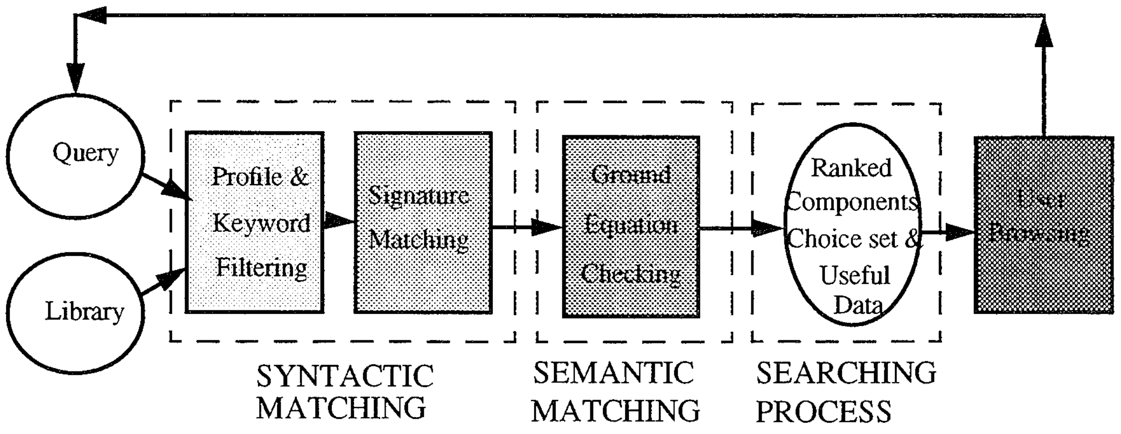


Figure 3. An Organization Model for Software Component Search

The implementation of the Multi-level filtering retrieval process is separated by three parts: the graphical user interface, the retrieval process kernel, and the data storage. This thesis is part of a joint-research project with two other CAPS team members; Doan Nguyen who made the major contribution of the retrieval kernel[Ref. 31] and Tuan Nguyen who dealt with populating the database and the specification axioms transformation[Ref. 32]. This thesis concentrates on the front-end graphical user interface of the retrieval module and the inner data storage design. The final payoff of the joint research was an novel application for accessing the software base within CAPS. It provides a significant improvement over current release for accessing the software base.

The retrieval kernel of the Multi-level filtering retrieval process is mainly composed of three parts: Syntactic Matching Filter, Semantic Matching Filter, and Searching Process. The query specification is entered through the graphical user interface(GUI) and sent into these three modules which then search the data repository for candidate components. The results are then displayed by the interactive browser so that the user can make a selection decision. The whole process is iterated until the user gets the desired component. Once a target has been reached, the software reuse work has been accomplished. The next three sections will discuss the Syntactic Matching Filter, Semantic Matching Filter and Searching process in more depth.

A. SYNTACTIC MATCHING FILTERING

Syntactic matching filtering is based on non-behavioral information about components, such as keywords and interface declarations. Here we use two levels of syntactic filtering. The first level uses keyword and profile matching filters. By using these filters, the search module computes the indexes which partition the software library. These partitions contain the candidate components. This filters speed up signature matching, and is discussed later. The second level is signature matching filter, which finds the maps that translate the type and function symbols of the query into the corresponding type and function symbols of the candidate components. Later on, the search module will compute Profile and Keyword Matching Ratio values for each candidate component and then combine them with each component's Signature Matching Ratio. The final combined value is referred to as KPS which stands for the product of Keyword, Profile, and Signature Matching Ratio[Ref. 31]. The matching filters mentioned above need further elaboration due to their importance and are discussed below.

1. Keyword Matching

Even though keyword matching has a measurement trade-off problem(discussed in section II.B.2.a), it is still a useful method for multi-level filtering. It is easy to use, inexpensive to implement, and good for indexing components. However, we have to use keyword filtering carefully with a limited number of general keywords that are controlled by a system administrator. Keywords describe categories of components and their relationships to other components. Sample categories might be data structures, mathematical functions, search routines, and navigation functions.

[Ref. 31] proposed the following measurement function to indicate how close the keywords of a query, Kw_Q, are to the keywords of a component, Kw_M:

$$KeywordMatchRatio(Kw_Q, Kw_M) = |Kw_Q \cap Kw_M| / |Kw_Q| \quad .$$

This function measures recall. The numerator represents the number of relevant retrieved keywords while the denominator represents the total number of relevant keywords in a query.

2. Signature Matching

This section introduces and illustrates the basic concepts of signature matching, under the assumption that there are no subsort relations; the more complex situation when a sort[1] set S has subsorts (i.e., a partial order containment relation $\leq$ on data types). We assume that each component M in the library has an associated algebraic specification T_M of the form:

$$(S', \Sigma', E'),$$

where (S', Σ') is a signature with a set S' of sorts and a set Σ' of functions whose arguments and results have sorts in S', and where E' is a set of equations stating properties that the functions in Σ' should satisfy.

We also assume that query, Q, is an algebraic specification of the form (S, Σ, E) where these symbols mean the same as T_M. The following illustrates these notions, using the notation of **OBJ3**[2]; definitions for signature, specification, etc.

Example 1 The algebraic specification for a module defining list of identifiers in our library might have a sort set S containing the sorts Id, List, and Bool, and a signature Σ of functions consisting of the empty list (denoted by nil), an append operation *, and a function to test whether an element is in the list, with the following syntax:

```
sorts Id Bool List
 op nil  :           -> List .
 op _*_  : Id List -> List .
 op _in_ : Id List -> Bool .
```

According to OBJ3, the equations in the specification might be:

```
I in nil = false .
```

1. It means "type". This thesis will use the words "sort" and "type" interchangeably, and will also use the words "function" and "operation" interchangeably.
2. OBJ3, an algebra specification language, which can accurately describe the semantic behavior of software components.

I in (I' * L) = if (I == I') then true else I in L fi .

We assume that the library has a set of **basic sorts** for commonly used types like booleans, identifiers, integers, and floating point numbers; that the names of these basic types and their associated basic operations are identical in the specifications and in the code; and that the library modules and the algebraic specifications for the basic types also have the same names.

Definition 1 Given two signatures (S, Σ) and (S', Σ'), a **permutative signature map** V: $(S,\Sigma) \rightarrow (S',\Sigma')$ consists of injective functions $V: S \rightarrow S'$ and $V: \Sigma \rightarrow \Sigma'$ such that for each function symbol $f: s_1 \ldots s_n \rightarrow s$ in Σ, there is a permutation π such that $V(f)$: $V(s_{\pi(1)}) \ldots V(s_{\pi(n)}) \rightarrow V(s)$ is a function symbol in Σ'. A **partial signature match** V: $(S, \Sigma) \rightarrow (S', \Sigma')$ is a permutative signature map $V: (S_o, \Sigma_o) \rightarrow (S', \Sigma')$ where (S_o, Σ_o) is a subsignature of (S, Σ); it is **total** if $(S_o, \Sigma_o) = (S, \Sigma)$.

The assumption that V is injective on both sorts and operations is reasonable if there is no subsort relations, because otherwise the user would be asking for two or more things that are not actually different.

Definition 2 Given a library and a query $Q = (S, \Sigma, E)$, the **signature choice set** for Q consists of all signature matches $V: (S, \Sigma) \rightarrow (S', \Sigma')$ where $T_M = (S', \Sigma', E')$ is the specification of some component M, and where each match V is the identity mapping when restriction to the set of basic types and their basic function symbols. Note that the semantic information in the equations is ignored in syntactic matching. To simplify notation and make explicit the module specification associated with a signature match, we may write $V: Q \rightarrow T_M$ for a signature match $V: (S, \Sigma) \rightarrow (S', \Sigma')$ such that $T_M = (S', \Sigma', E')$ is the specification of a component M.

The more complex situation when S has subsorts. For example, a partial order relation $\subseteq$ is illustrated in the following 12 sort sets:

Suppose that we have 12 sort sets $V_1, V_2, \ldots, V_{12}$, where

$V_1 = \{ A \}$

$V_2 = \{ A, B \}$

$V_3 = \{ A, B, C \}$ $V_4 = \{ A, B, D \}$ $V_5 = \{ A, B, E \}$

$V_6 = \{ A, B, C, D\}$ $V_7 = \{ A ,B ,C ,F \}$ $V_8 = \{A, B, C, D, E, F \}$

$V_9 = \{A, B, E, G \}$ $V_{10} = \{A, B, D, E \}$

$V_{11} = \{A, B, C, D, E, G \}$ $V_{12} = \{A, B, C, D, E, F \}$

Then we have the relation $\subseteq$ as follow:

$V_1 \subseteq V_2$

$V_2 \subseteq V_3$ $V_2 \subseteq V_4$ $V_2 \subseteq V_5$

$V_3 \subseteq V_6$ $V_3 \subseteq V_7$ $V_3 \subseteq V_8$

$V_4 \subseteq V_6$ $V_4 \subseteq V_8$ $V_4 \subseteq V_{10}$

$V_5 \subseteq V_8$ $V_5 \subseteq V_9$ $V_5 \subseteq V_{10}$

$V_6 \subseteq V_{11}$

$V_7 \subseteq V_{12}$

$V_8 \subseteq V_{11}$ $V8 \subseteq V_{12}$

$V_9 \subseteq V_{11}$

$V_{10} \subseteq V_{12}$

Figure 4 describes the Hasse Diagram of these 12 sort sets. (A review of Partial Ordering Relations and Hasse Diagram concepts is given in Appendix A)

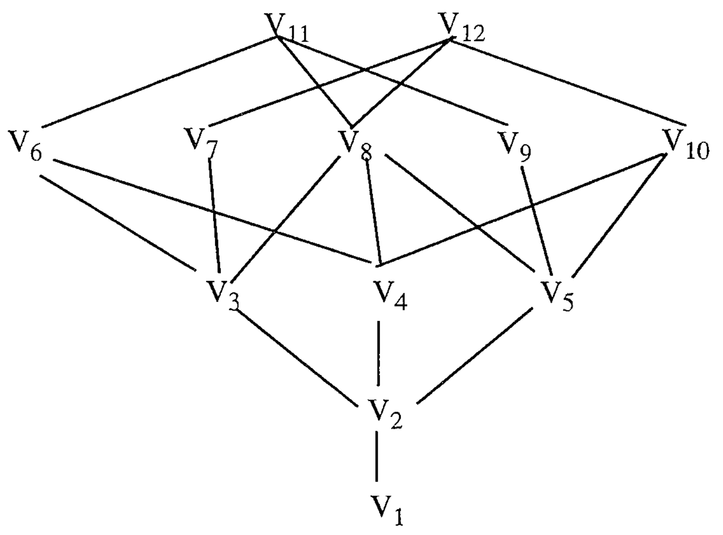


Figure 4. A Hasse Diagram for the Signature Map Refinement Relation $\subseteq$

Note that only the maximal elements V_{11} and V_{12} in this partial ordering are of interest. All the others will be strictly less useful. This provides a pruning criterion for signature matching.

3. Profile Matching

The computations for signature matching would be very expensive if it were necessary to try all possible ways of pairing the functions and sorts of queries with those of components. It is therefore highly desirable to cut down the search space. This can be done. For example, if a query has a function $f\colon\ AAB \to B$ and a component has a function $g\colon\ ABC \to D$, then it is obvious that these functions cannot match, because their arguments have different sort patterns; there is no need to compute all possible sort maps to draw this conclusion.

The purpose of profile matching is to speed up signature matching. Profile matching is actually an efficient approximation of signature matching. A profile is a sequence of numbers that describes how the sorts associated with an operation are organized. We can quickly determine whether two given operations could possibly match by comparing their profiles, and hence quickly identify query operations and test cases that will necessarily fail. Profile matching uses pre-computed profile indexes to partition the library, and profile values are used as search keys in seeking components having suitable signatures.

We now introduce some concepts to help us define profiles. The **sort groups** of an operation are bags (i.e., multisets) consisting of two or more sort occurrences from the rank (i.e., the argument plus value sorts) of the operation that are related under the relation $\equiv$, which is the transitive-symmetric closure of the ordering $\subseteq$ on sorts. The **unrelated sort group** is the bag (actually a set) of all sort occurrences that are not in any sort group.

Definition 3 The **profile** of an operation is a sequence of integers, defined as follows:

- The first integer is the total number of occurrences of sorts.
- If the total number of sort groups, N, is greater than 0, then the second to $(1+N)^{th}$ integers are the cardinalities of the sort groups, in descending order.
- The $(2+N)^{th}$ integer is the cardinality of the unrelated sort group.
- The $(3+N)^{th}$ integer is:

  0 if the value sort is different from any of the argument sorts; and

  1 if the value sort belongs to some sort group.

A signature map can relate two operations only if they have the same profile (i.e., the same number of sort occurrences, the same number of sort groups, the same sort group cardinalities, and the same unrelated sort group cardinality).

Some sample profiles are shown in Table 1, where A, A', B, C, E, F, G are sorts, and A' is a subsort of A:

Operation	Profile
$\rightarrow A$	110
$EF \rightarrow G$	330
$AA \rightarrow B$	3210
$ABBCA \rightarrow C$	622201
$CCBAA \rightarrow B$	622201
$CCBAA \rightarrow A$	63211
$CCBAAA' \rightarrow A$	74211

Table 1: Some Operations and their Profiles

For the operation "$\rightarrow A$":

(1). There is only one occurrence of sort, therefore the first integer is 1.

(2). Since there is no sort group occurs in this case($N = 0$), we skip the cardinality calculating rule for the sort groups.

(3). The cardinality of the unrelated sort group is 1, so the $(2 + 0)^{th}$ integer is 1.

(4). The value(range) sort is different from any of the argument(domain) sort, therefore the $(3 + 0)^{th}$ integer is 0.

The final profile of this operation is 110.

For the operation "$ABBCA \rightarrow C$ ":

(1). There are six occurrences of sorts, therefore the first integer is 6.

(2). By the definition of sort groups, there are three sort groups *AA, BB and CC* in this case($N = 3$), hence the second to $(1 + 3)^{th}$ integers are the cardinality of each of them in decreasing order, which are 2, 2, 2.

(3). Since there is no unrelated sort groups in this case, the $(2 + 3)^{th}$ integer(the cardinality of the unrelated sort groups) is 0.

(4). The value(range) sort "A" belongs to one of the argument(domain) sort groups, therefor the $(3 + 3)^{th}$ integer is 1.

The final profile of this operation is 622201.

B. SEMANTIC MATCHING FILTERING

The choice set of signature matches for a query can be further narrowed by semantic matching filter. Under certain reasonable assumptions, this can be done efficiently by checking satisfaction of certain semantic conditions that are necessary for any correct answer to the query.

We discuss only unparameterized specifications here. In addition, we assume throughout that the specifications associated with components have equations that are Church-Rosser and terminating. This means we can apply the equations from left to right to simplify a term to a unique simplest possible form, called its canonical form, which can be regarded as the result of evaluating the term. For example, the canonical form of the term.

a in (b * (a * nil))

using the equations in Example 1 is true.

The semantic validation procedure takes ground equations $t=t'$ from the query specification Q and tests them for satisfaction in the candidate specifications. *Ground equations*, that is, equations whose terms have no variables, are particularly useful here, because any ground equation provable from an equational theory Q is satisfied by the standard model of such a theory, i.e., the initial algebra T_Q of Q, so that those equations are also satisfied under the initial (standard) interpretation of Q.

This is important because either the query Q or the component specifications T_M may have an initial interpretation, so that proving the semantic correctness of a signature match V: $Q \rightarrow T_M$ could require complex theorem proving to check inductive consequences. A special property of ground equations is that their translations $V(t) = V(t')$ must be provable from T_M in order for V to be correct, regardless of whether Q and T_M are interpreted initially or loosely. Therefore, we can use them under either interpretation to further restrict the search for correct answers to the query Q. The great advantage of ground equation is that we can automatically settle the issue of whether $V(t) = V(t')$ is provable from T_M by comparing the irreducible forms of $V(t)$ and $V(t')$ after rewriting them with the equations in T_M, since T_M is assumed to be Church-Rosser and terminating. This results in an efficient decision procedure for behavior queries expressed in this form.

The following describes how to rank members of the choice set based on semantic filtering.

Since many signature matches for a query might be found in a large library, it is important to narrow the search by using whatever semantic information is available at each filtering stage, either from the specification T_M, the compiled component M, or both. For this, a sound way of ordering the matches according to their relative degree of semantic correctness is needed. We define below two measures that can be used for this purpose. These measures assign a value to each pair (V, I_V) consisting of a signature match V for the query Q, and whatever information I_V is available at that stage about equations that have passed or failed the semantic checks. Such measures can be used independently or in combination to define choice sets.

Given a match V, the information I_V available for it may be either syntactic or semantic. Syntactic information will include the functions in the query's signature for which the match is defined and their translation under such a match. Semantic information will include the results of checking correctness of ground equations after translating them through V. For each such ground equation and match V three things can happen:

- The translated equation is well defined and, after reducing each side to normal form using the equations in the specification of the module matched by V, yields an *identity*; therefore this equation has *succeeded* for this match;
- The translated equation is well defined and, after reducing each side to normal form using the equations in the specification of the module matched by V, yields an equation whose two sides are different; therefore this equation has *failed* for this match;
- The equation could not be translated, because the match V was undefined for some of the functions appearing in the terms of the equation; this is also a kind of *failure*.

Note that we can associate each equation with a function symbol, namely the top function symbol of its left side. Therefore, we can assume that the semantic information in I_V is organized by function symbol so that for each such symbol f in the query's signature we have a set of ground equations for it, and information about their success or failure for the match V.

The first measure μ assigns to each pair (V,I_V) a real-valued function $\mu_{(V,I_V)} : \Omega \rightarrow \Re$, where Ω is the signature of the query Q , defined as follows:

- If $V(f)$ is undefined, then $\mu_{(V,I_v)}(f) = 0$.
- $V(f)$ is defined but has no ground equations associated with it, then $\mu_{(V,I_V)}(f) = 1$.
- Otherwise, $\mu_{(V,I_V)} = \frac{Success(f)}{Equation(f)}$, where $success(f)$ is the number of successful checks for ground equations $t = t'$ with t having f as its top function symbol reported in I_V , $Equations(f)$ is the total number of such equations.

Consider now two different matches V and V' for the same query Q, and suppose that the same ground equations from Q have been tried for V and for V'. Then we can compare the degree of success of these matches on an operation by operation basis. If for each operation symbol f we have $\mu_{(V,I_V)}(f) \geq \mu_{(V',I_W)}(f)$, then V is an altogether better match than V'. The ideal situation is of course when we find a match that is better than all other matches in exactly this sense. However, such an absolutely better match may not exist in the library and we may only get matches that are *maximal* in their degree of success, that is, no other match is better than them for all functions. This can happen when a query is large enough that two or more parts of the required functionality are available, but their combination is not. In such a case we will find several maximal signature matches that are each best for some fragment of the functionality, but are incomparable among themselves under the μ ordering. An appropriate environment could use this information to help the user synthesize an optimal combined component
out of actual components whose corresponding signature matches have maximal μ -measures.

The second measure *SemanticMatchRatio* is cruder. It is obtained from the first by assigning to each pair (V,I_V) a real number defined by the equation:

$$SemanticMatchRatio(V, I_V) = \sum_{f \in \Omega} \mu_{(V, I_V)}(f) \qquad ,$$

where Ω is the signature of the query Q.

Once we obtain a SemanticMatchRatio of a component, we can compute the overall *ComponentRank.* Since Semantic Matching Ratio is the most significant information and derivable from Profile Matching and Signature Matching, it is important that we order the component base on this. Keyword Match Ratio is the second important piece of information due to how well a user can specify the keywords. The ComponentRank is then a 2-tuple of matching ratios with lexicographic ordering and is defined as follows:

- A component which has a higher SemanticMatchRatio is ordered first. ComponentRank would be included its SemanticMatchRatio and KeywordMatchRatio.
- Any components which have the same SemanticMatchRatios, then a component with a higher KeywordMatchRatio will be used to order the components. Again ComponentRank would be included its semanticMatchRatio and KeywordMatchRatio.

C. SEARCHING PROCESS

After MLFRT did the syntactic matching filtering and the semantic matching filtering, it needs to retrieve the candidate components from the physical data storage, which will be discussed in chapter VI. There should be two important data that reside on the physical data storage. First, the data used for matching. Second, the data used for importation to a user's prototype. The data used for matching are specified by item (a) and (b) below. Once a component is selected by the user, the data used for importation to user's prototype is specified by item (a), (c), and (d). Therefore, we organize the physical file representation of a software base component as follows:

(a) A file containing a PSDL specification, for the translation and scheduling for a prototype using the component.

(b) A file containing a OBJ3 specification, used for both signature and semantic matching.

(c) A file containing an Ada specification, which is imported to become an atomic operator's Ada specification in a prototype, after some modifications.

(d) A file containing an Ada body, corresponding to the semantic part. It is imported to a prototype, to become an atomic operator's Ada body, after some modifications.

This organization supports traditional graph search algorithms, such as depth first search, to implement DBMS operations such as initialize, delete, add, and retrieve. There could also be a file for compiled code, which could replace the use of OBJ3 for some queries.

The following algorithms find the candidate components in a software library:

```
Algorithm 1 FindCandidateComponents
Input:
    Q = (S,E).
    KwQ = {Query Keyword Ids}.
    G = (P,R) where P = {Partitions} and R = {Inclusive relations}.
    ComponentLookupTable = Array of 2-field records: Component Keywords and
    Component Pointer.
Output:
    CandidateTable = Array of 3-field records: Component Id, Keyword Matching Ratio, and
    Profile Matching Ratio.
    Invalid Operations = {Invalid Operations}
    (0) Initialize CandidateTable to empty.
    (1) Compute profile value, p, for each operation in Q.S.
    (2) For each profile p, verify against the bottom nodes in G. If Freq(p) =0, then store p in
        Invalid Operations.
    (3) For each ground equation i in Q.E, identify its associated profiles using the
        correspondinn operation's profile values found in step (1). Store these associated
        profiles in called Gp(i). (Note: Gp(i) is an array of multisets).
    (4) Let N be the set of starting nodes that contains every node indexed by profile multi-set
        of size one that is included in the query profile.
    (5) For each n in N, call DepthFirstSearchForward.
    (6) Report invalid operation(s) in Invalid Operations.

Algorithm 2 DepthFirstSearchForward
Input:
    G = (P,R) where P = {Partitions} and R = {Inclusive relations}.
    v = An element of P.
    CandidateTable = Array of 3-field records: Component Id, Keyword Matching Ratio, and
```

```
        Profile Matching Ratio.
        Gp = {Gp(i), where Gp(i) is a multiset of profile in a ground equation i in Q.E}
        ComponentLookupTable = Array of 2-field records: Component Keywords and
        Component Pointer.
Output:
        CandidateTable = Array of 3-field records: Component Id, Keyword Matching Ratio, and
        Profile Matching Ratio.
        (1) Mark v visited. Assign Pv with Partition v's profile multiset.
        (2) For each Gp(i) in Gp, If Gp(i) contains Pv then
                Calculate the Profile Match Ratio.
                For each component pointer at node v do
                        Index to an element in ComponentLookupTable.
                        Calculate the Keyword Match Ratio.
                        Store Profile and Keyword Match Ratios and
                        ComponentID in CandidateTable.
        (3) For each vertex n adjacent and above v do
                If n not visited then DepthFirstSearchForward.
```

D. CONCLUSIONS

The multi-level filtering retrieval process(MLFRP) must be further evaluated and implemented for more practical components that can easily map to the real world. This thesis concentrates on the front-end graphical user interface of this MLFRP and the physical data storage design which will be discussed in the next two chapters. Chapter VI discusses the conceptual design for the physical data storage of the reusable software components. Chapter V discusses the design issue and the implementation detail for the graphical user interface of MLFRP.

IV. DATA STORAGE DESIGN

Up to now we have discussed the searching process and multi-level filtering retrieval technique(MLFRT) for the CAPS software database. The next issue addressed in this thesis is making a conceptual design of a physical data repository for the reusable software components.

A. INTRODUCTION

As we mentioned in chapter II, the searching process, retrieval technique and the design of data repository are tied together in the research of software reuse. The more cohesive the design integrity, the better efficiency the final retrieval performance. [Ref. 31] has made a small scale software base mock up which stores 12 typical data structure components in unix based files. The experiment in Nguyen's work showed that the combination of the multi-level filtering retrieval technique(MLFRT) and the DFS searching algorithm does work. It also showed that the MLFRT is the best candidate for searching the database because of its high recall, high precision, and reduced search time. However, this result has been validated only for specific data structure types. It must be further evaluated and implemented for more practical components that can easily map to the real world. This chapter discusses the conceptual design of the physical data storage of the reusable software components. We discuss several design issues based on the previously discussed searching process and the MLFRT. Nevertheless, the separate individual implementation does not effect the integrity of this conceptual design.

B. REQUIREMENTS ANALYSIS

The data storage of the CAPS software base is broken into three sets of requirements. It has basic requirements which cover background requirements, system requirements which are dictated by CAPS, and functional requirements which necessitate traditional DB support functionality.

1. Basic Requirements

- Based on the requirements of the MLFRT, the data storage should store four pieces of source code for each component. They are the Ada specification, Ada body, PSDL, and OBJ3 code.
- The database schema design should support all the defined terms in the MLFRT module, such as partial ordering, syntactic matching, signature matching, etc.
- Because of the dynamic length of the data entries, an object-based database system

is required.

- Because of the highly inter-related attribute characteristics and the stringent requirements for searching performance, an object-oriented database system is required.
- The software component repository should be able to be maintained and manipulated without connecting with any searching engine or any MLFRT module and therefore a direct manipulation interface is required.

2. System Requirements

Because of the constraints of CAPS, the software component repository should meet the following system requirements:

- The repository must be accessible in the unix based workstation, such as Sun or Solborne or the compatible platforms.
- The repository must be accessible in X-windows or the compatible windows environments.

3. Functional Requirements

- The software component repository should support the basic database manipulation functionality, such as append, delete, modify, browse, backup, etc.
- The database schema should accommodate or at least consider the flexibility of the multi-type attributes of the reusable software components, such as data-structure, GUI metaphors, communications, etc.

C. DESIGN STRATEGIES

Database design is always an arguable issue in that many different designs can result from the same requirements. Implementations too can vary with the same design. This feature is even worse in the object-oriented database design and programming area. In this section we explores two design strategies which are based on the previous requirement analysis. The pros and cons for each strategy are also discussed.

The first strategy is used in the mock up in [Ref. 31]. Let's call it the semi-automatic facilitative software component repository. This strategy picks a proper object-oriented database engine,[1] such as ONTOS, which will be discussed later, to be the storage mechanism for the software components. The purpose of this database engine is to deal with the simple database I/O process which means we only regard it as a data storage engine(just for storing/write and retrieval/ read data). In such a strategy, no complicated search algorithm will be involved in the Database Manipulation Language(DML) of the database engine, which in this case is C++. This means we do not have to put the complete database schema into the database because all the attribute information is kept in another small fixed-length, text-based assistant(companion) database.[2] All searching algorithms must be applied to this auxiliary database. Once the key information of the candidate components have been gathered(such as physical file name, component-ID number, etc.), the software base system passes the key information into the real object repository. From this point it connects to the DML for accessing the I/O process of the desired components.

The advantages of this strategy are as follow: First, the reduced involvement with the database simplifies data storage and therefore helps keep the overall task of the DBMS much simpler. Second, it isolates the implementation tasks between the searching and retrieval process and the data object manipulating actions. It helps facilitate better database transportability for the future system upgrading or porting activity. Third, because of the implementation-isolated feature, the searching process has a distinct, uncoupled refinement space. The DML independent characteristic modularizes the development of the searching process resulting in more defined space for the future research. It also meets the flexible open architecture system design principle. However, the simpler utilization of the database repository also have several disadvantages. First, in the long run, the DBMA(Database Management Administrator) of the software base has to manage two separate databases which it decreases system integrity and increases the overall system load. Second, no matter how perfect the system analysis is, the interface between the auxiliary database and the major data repository will always be a problem. Some overlapped manipulating mechanism might be developed on both sides due to the duplicated requirement analysis. Third, it increases the system un-interoperability hazard because of the extra connection task, let along handling the speed mismatch between different languages(i.e., one for the searching process and one for the DML of the database engine).

The second strategy, which is still in the conceptual design stage, grants the database engine a complete manipulating capability over the software components. In this strategy, a com-

1. Engine means it meets or it supports most of the basic database facilitating functionality, yet it requires user written utilities to make it work instead of the intuitive plug and run activities.
2. It might be a unix-based-file or represented by any other formalism such as a sequential file or a direct access file.

plete data object schema[3] analysis needs to be done first and then will be defined with the Database Definition Language(DDL) in the selected object-oriented database engine. The SBGUI(Software Base Graphical User Interface), discussed in chapter V, directly passes the query information, which is taken from the user, into the database engine to allow the DML to conduct the searching process.

Basically, the advantages of the second strategy are the opposite of the first strategy. First, it has a clean and easier architecture. As in most current existing database systems, a system architecture like this usually helps make maintenance much easier. Second, it decreases the uninteroperability hazard between the SBGUI and the real data repository. Third, it reduces the Database Management Administrator(DBMA)'s overall work load significantly compared to that of the first strategy. Fourth, there is less chance for analysis mistakes between the SBGUI and the OODB due to the clear roles they play.

However, the disadvantages are not so obvious. First, the implementation is harder because of the dependency of the DML. Since all search process and MLFRT has to be implemented by the DML, the developer must become intimately familiar with the DML of the OODBMS. Second, the more DML involved, the less database transportability available. The DBMS dependency feature also restricts the future system upgrading capability. Third, it has lower system flexibility. Once the search algorithm is modified, the DML has to be rewritten. Figure 5 illustrates the concept of the two design strategies.

3. In C++, it is called Class.

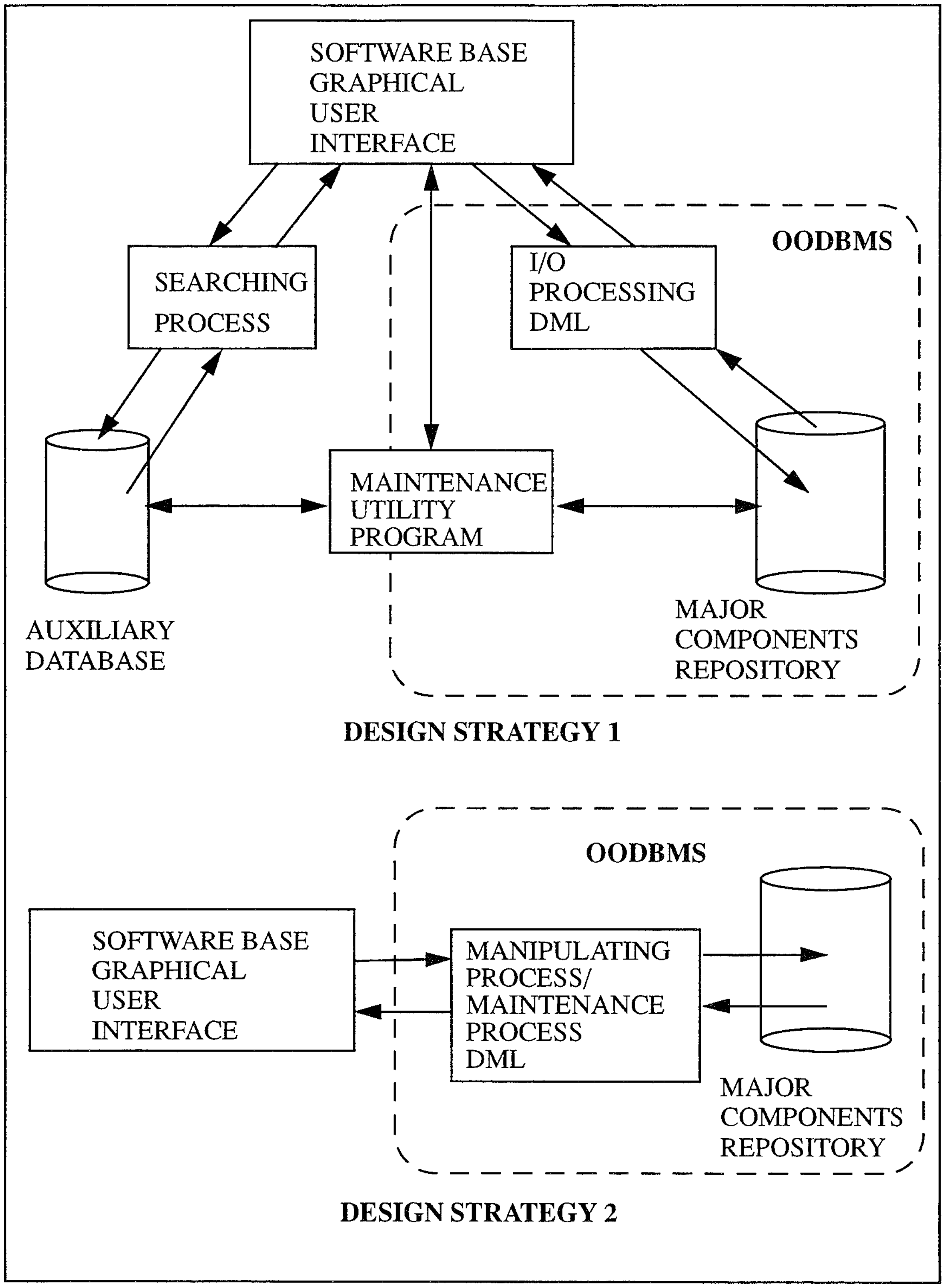

SOFTWARE BASE GRAPHICAL USER INTERFACE
SEARCHING PROCESS
I/O PROCESSING DML
OODBMS
MAINTENANCE UTILITY PROGRAM
AUXILIARY DATABASE
MAJOR COMPONENTS REPOSITORY
DESIGN STRATEGY 1
OODBMS
SOFTWARE BASE GRAPHICAL USER INTERFACE
MANIPULATING PROCESS/ MAINTENANCE PROCESS DML
MAJOR COMPONENTS REPOSITORY
DESIGN STRATEGY 2

Figure 5. Two Design Strategies for Software Database

D. EVALUATION

Some evaluation work has been done for these two strategies. McDowell used the second strategy to create a small database with operator component objects and abstract data type objects in his master thesis[Ref. 33]. Dolgoff[Ref. 30] improved his work with a more sophisticated consideration of the physical schema definition and more precise matching process.[4] Both of them used the ONTOS OODBMS to be the database storage engine. The practical developing experience and the empirical performance told us that further improvement is getting more and more difficult because of the strong cohesive feature among the search process and the DML code.

The experiment in [Ref. 31] proved the MLFRT is the best candidate search process for accessing the software base. However, it used the first design strategy to reach the theoretical proof for the MLFRT technique. From a technical point of view, to use a companion database might be a good proposal for the Decision Support System(DSS) in some management research areas, yet it is not proper for implementing a mass software component storage system. CAPS software base research members spent a great deal of time attempting to find the best solution to solve the complex inter-related data representation between Ada and C++.

[Ref. 34] proposed a step through methodology which supports a smooth bridge control method and overcomes the integration obstacles between Ada and C++; where Ada is the major implementation language for CAPS and C++ is the DML of the ONTOS database engine. By using this method, the empirical implementation for the searching algorithm acts like that used in the first strategy. This means both the CAPS implementation language(Ada), and the DML for the ONTOS database engine(C++), are involved in the searching algorithm. However, the physical data storage design strategy acts like that of the second strategy, which means there is no companion database. The ONTOS data repository will be the only database, which helps the system integration and reduces the over all work load for the DBMA. Figure 6 illustrates the concept of the implementation by using the bridge control technique.

4. Scott Joel Dolgoff proposed an ordering matching process, which includes operator component match ordering and type component match ordering.

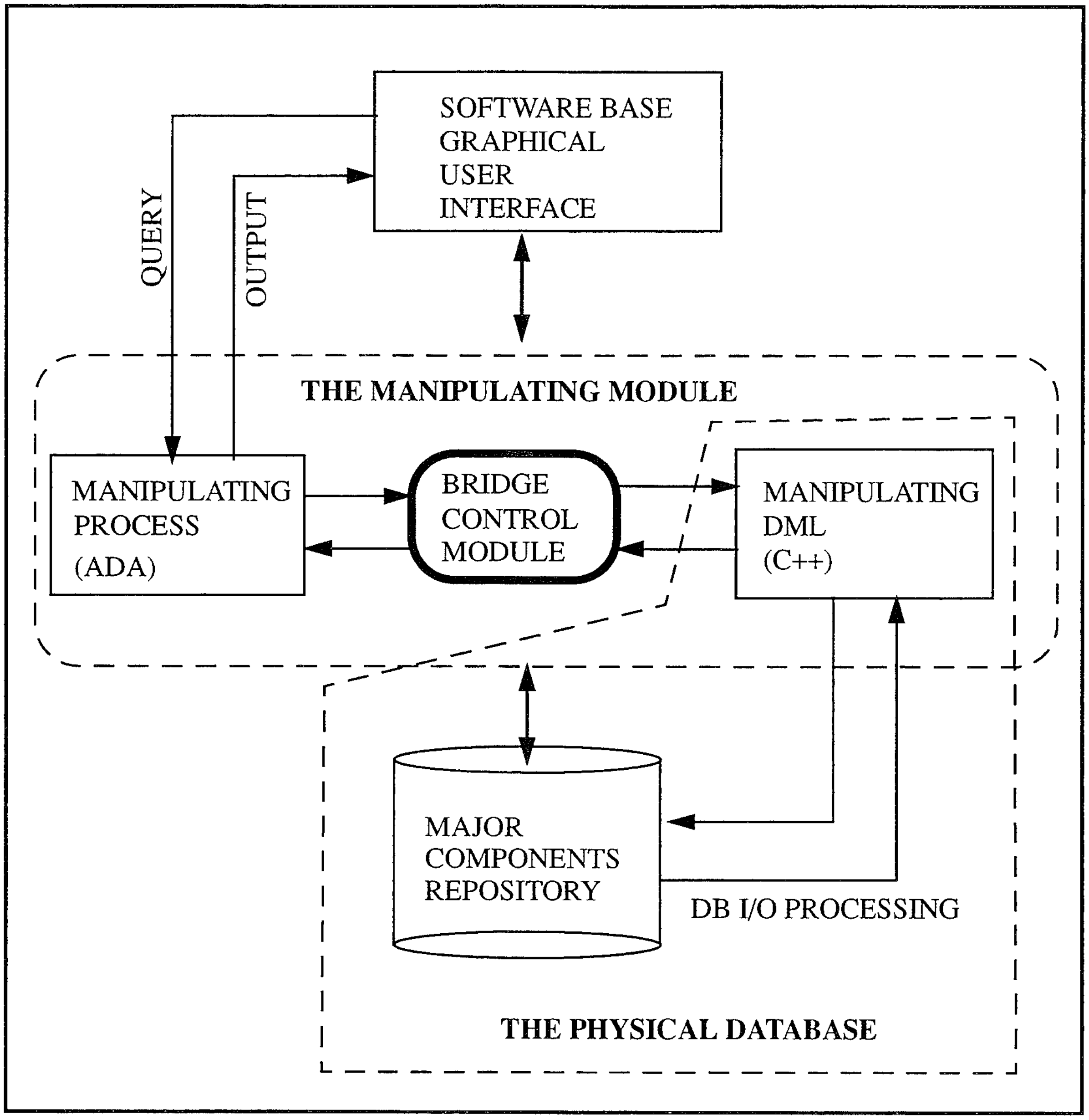


Figure 6. Practical Implementation Concept of the CAPS Software Database

E. IMPLEMENTATION DETAIL

1. ADT Design

Since this thesis concentrates on the software base while applying the multi-level filtering retrieval technique, the database schema must meet the requirements mentioned earlier(Chapter V.B) in order to properly support the database. The most important abstract data type(ADT) we have to define for the software base is the Component, which is the physical node that resides in the hasse diagram and represents a real software component. Figure 7 illustrates the Component class definition diagram.

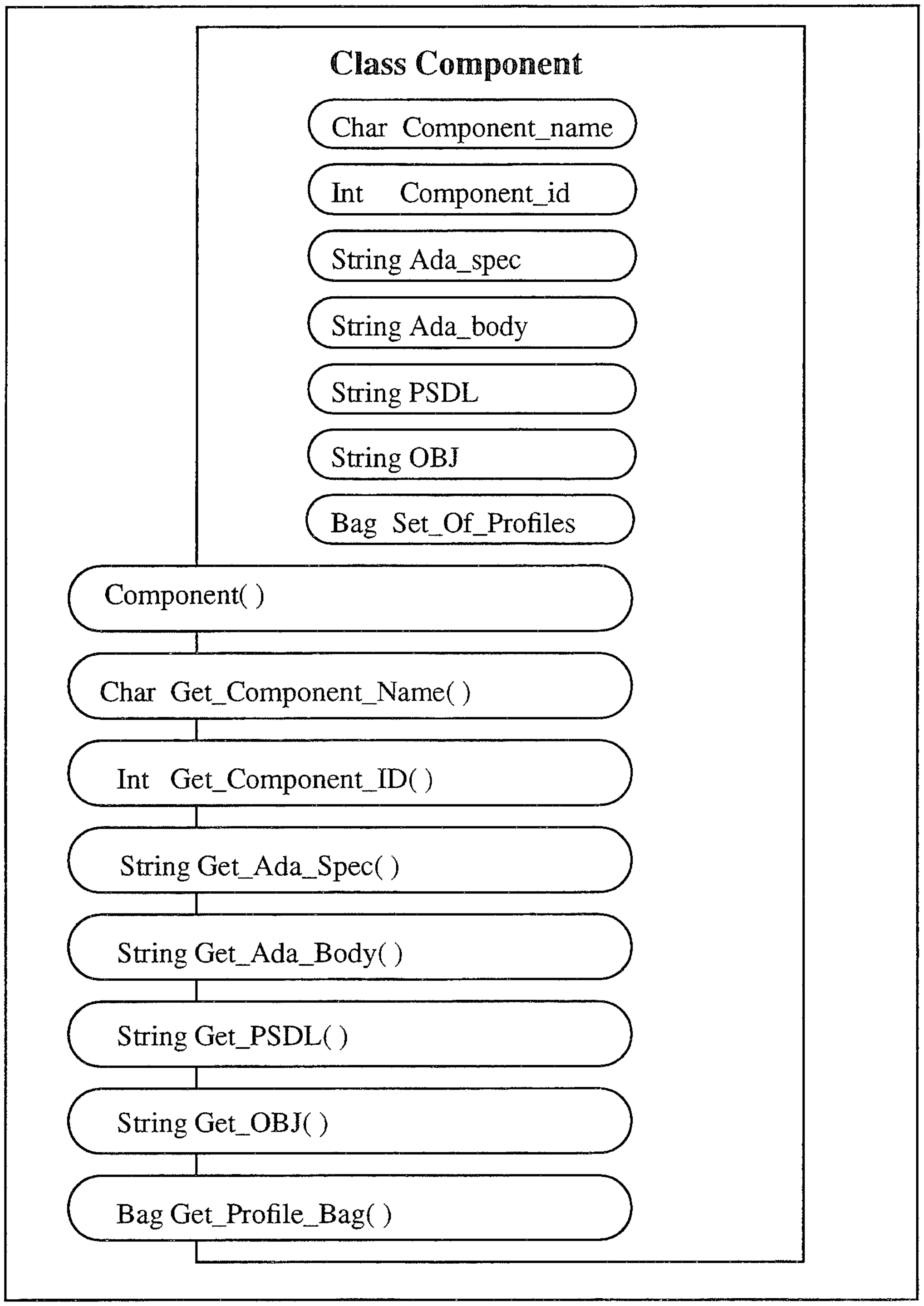


Figure 7. Component Class Definition

2. Bridge Control between Ada and C++

Ada is used by CAPS for developing the rapid prototypes and the target application. However, currently there is not a OODBMS which supports Ada. We found out that ONTOS is an adequate database engine that meets all the requirements of the software base. The only problem is that it is based on C++, which raises an issue for the implementation; how to interface the communication between Ada and C++?

[Ref. 30, 33] had tried to use system calls to solve the inter-communication problem between Ada and C++ in their software base mock up experiments. Empirical experience showed that implementation in this way is quite dangerous. First, by using system calls, it is not easy for the caller to facilitate all of the functionalities in the called side. Second, as this would be a highly interactive system, which will continue to use system calls (Ada calling C++ or C++ calling Ada), many errors will occur due to the parameter passing and the different data representation between the two languages. Many unexpected results could happen, e.g. constraint errors due to the different conventions of Ada and C++; their arrays, strings, numerators, to name a few. Such experience showed that, while the system calls might be a good method for triggering another unrelated executable program within a certain application, it is not suitable for a highly inter-communicated system like the situation in our CAPS software database.

A better way to solve this problem is to allow the software communicated with each other via the C++ interface package in Ada (Interface.C), which supports importing C functions into Ada and exporting Ada subprograms to C++[Ref. 35]. [Ref. 34] has successfully developed a bridge control technique in the on-going CAPS Design Database research by using the C++ interface library in Ada. This thesis has also successfully developed a software base mock up which meets all the MLFRT constraints and requirements by applying the same bridge control technique. The sources code of the mock up is in Appendix B. Figure 8 illustrates a sample usage of the pragma in Ada for calling C++ functions.

```
procedure ada_c_open_database(ddb : in c_string) ;

 pragma interface(C,ada_c_open_database);
 pragma interface_name(ada_c_open_database, "_ada_c_open_database__FPc");
```

Figure 8. Sample Ada Interface Pragma

The key point of the bridge control technique is that the user has to define the same data types and data operators in Ada and C++ simultaneously, and referenced as mirror programs. Figure 9 illustrates the mirror programs in the mock up of the software base.

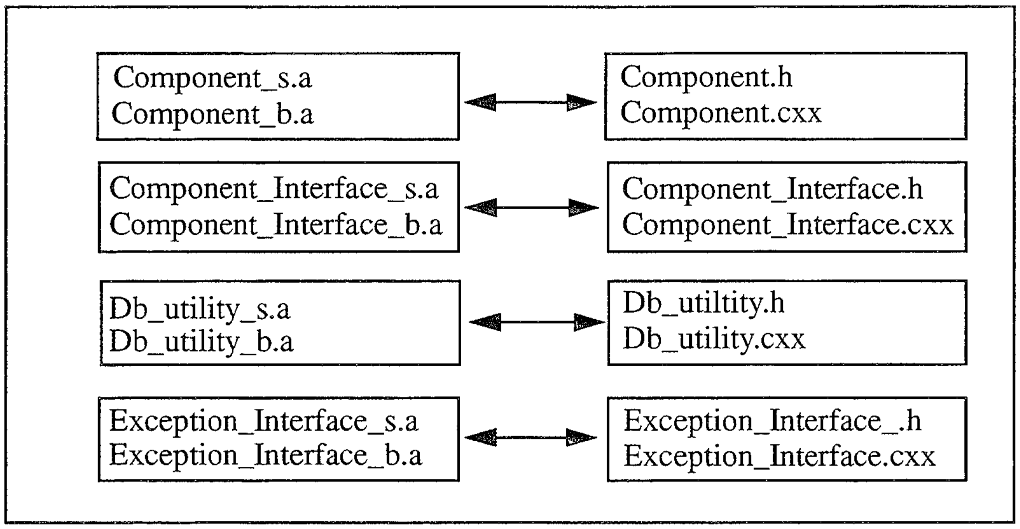


Figure 9. Mirror Programs in CAPS Software Base

Another important feature of the bridge control technique is that all the parameter passing activities are called by reference. By using such a strategy, the bridge control technique successfully skips all of the tiny data representation discrepancies between Ada and C++. It also reduces the potential hazard for communication errors. Figure 10 illustrates the concept of the calling module.

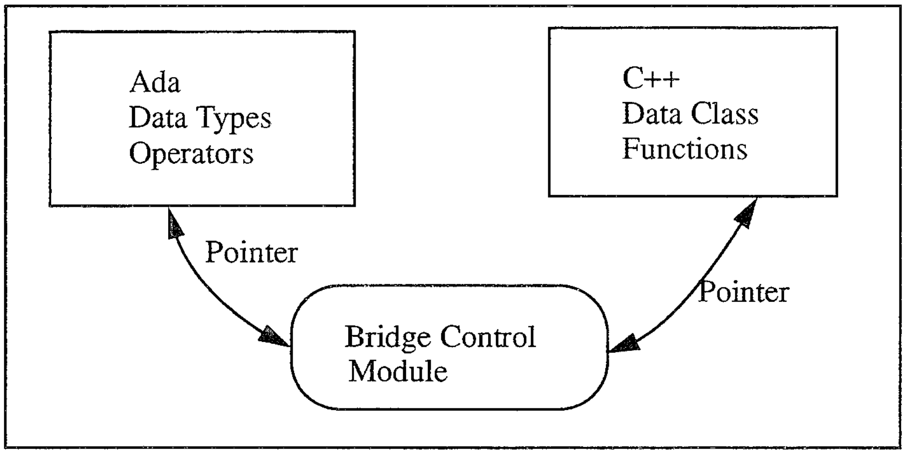


Figure 10. Calling Module of the Ada/C++ Bridge Control

3. Using ONTOS

ONTOS is a powerful object-oriented database management system(OODBMS), nevertheless, unlike the other relational database management systems, there is no Integrated Development Environment(IDE) to facilitate easy of use. This limitation, however, is common throughout all OODBMS(ex. Omnisciences OODBMS). Part of the reason is that the object oriented database has a feature of highly inter-related attributes. Because of this, users have to create the database template and define the schema by writing C++ code and then commit them into the database themselves. Therefore, a series of command line instructions are required for the processing. A quick ONTOS usage manual is in Appendix C.

V. GRAPHICAL USER INTERFACE

Because of leaping improvements in computer soft/hardware and network operating system, Graphical User Interfaces(GUI) are becoming more and more important. The purpose of a GUI is to speed up the software exploration operation, help user to sophisticatedly manipulate the designed functionality of the application, and support a user friendly interactive environment. Excellent software without a well-designed GUI is very limited and sometimes useless. It is like a Russian astronaut working with a US counterpart on the space shuttle. They both are from excellent space programs, but if they don't speak the same language, nothing is accomplished. The communication obstacle washes out the well-trained capability. GUI plays the same role of communication, only between users and software applications.

CAPS is a set of many individual tools. It includes the Syntax Direct Editor, PSDL editor, graphical editor, user interface editor, etc. To link all these tools together, a user friendly *tool interface* is needed, so that the user can effectively utilize the CAPS tool set. Thus, how to select a user interface tool and how to design the user interface itself becomes an issue. While this thesis concentrates on the CAPS software base and techniques to implement it, some attention must be focused on the interface to the software base in order for it to be as effective as possible. This chapter was designed to do just that.

A. REQUIREMENTS ANALYSIS

The first CAPS Software Base Graphical User Interface(SBGUI) was developed by Dogan Ozdemir[Ref. 9]. Later on, Scott Dolgoff[Ref. 30] made some improvement to it by providing the integration of software base operator components. This thesis makes two primary changes to their work. First, it includes the software base maintenance issue. Second, the design is based on the multi-level filtering retrieval algorithm. All interactive actions are driven by the multi-level filtering retrieval theory.

1. Basic Requirements

SBGUI should meet the following basic requirements:

- Must have high accommodation for a wide range of users from novice to expert users.
- Must provide on-line help.
- Must be easy to learn and easy to use.

- Must provide feedback for every action.
- Must provide an exit route for each level.
- Must provide enough menu driven information without on-line help and users manual.
- Must support message display and recovery handling when a mistake is made
- Must have consistency throughout.
- Must follow the 5 Principles[1] of Graphical User Interface Design.

2. Functional Requirements

The major functionality of the SBGUI should be separated into two parts: User's prospectus and Database Management Administrator(DBMA)'s Prospectus. Figure 11 illustrates the functional architecture analysis of the SBGUI(On-Line help and Exit functionality have been implemented but are not shown). From the users' point of view, the main concern is executing queries. Therefore, query is the only function shown in Figure 11 under the users' prospectus. Query functionality can be separated into Manual and Import. Manual means that the user has to input the query specification manually. Import identifies the capability of searching the software library based on the specification automatically generated from the SDE. When the query specification is accepted, it is then sent into the Multi-Level Filtering Retrieval module to search the data repository for candidate components. The results are then displayed by the interactive browser, so that the user can make a selection decision. The whole process will be iterated until the user gets the desired component. In the DBMA's point of view, software base maintenance work should include library creation, selection, deletion, and backup. After the DBMA selects a desired library, s/he should be able to append, delete, modify, and browse software components.

1. The 5 principles for Graphical User Interface design, proposed by Lewis & Riemann, are 1.The Clustering Principle. 2.The Visibility Reflects Usefulness Principle. 3.The Intelligent Consistency Principle. 4.The Color As a Supplement Principle. 5. The Reduced Clutter Principle[Ref. 36].

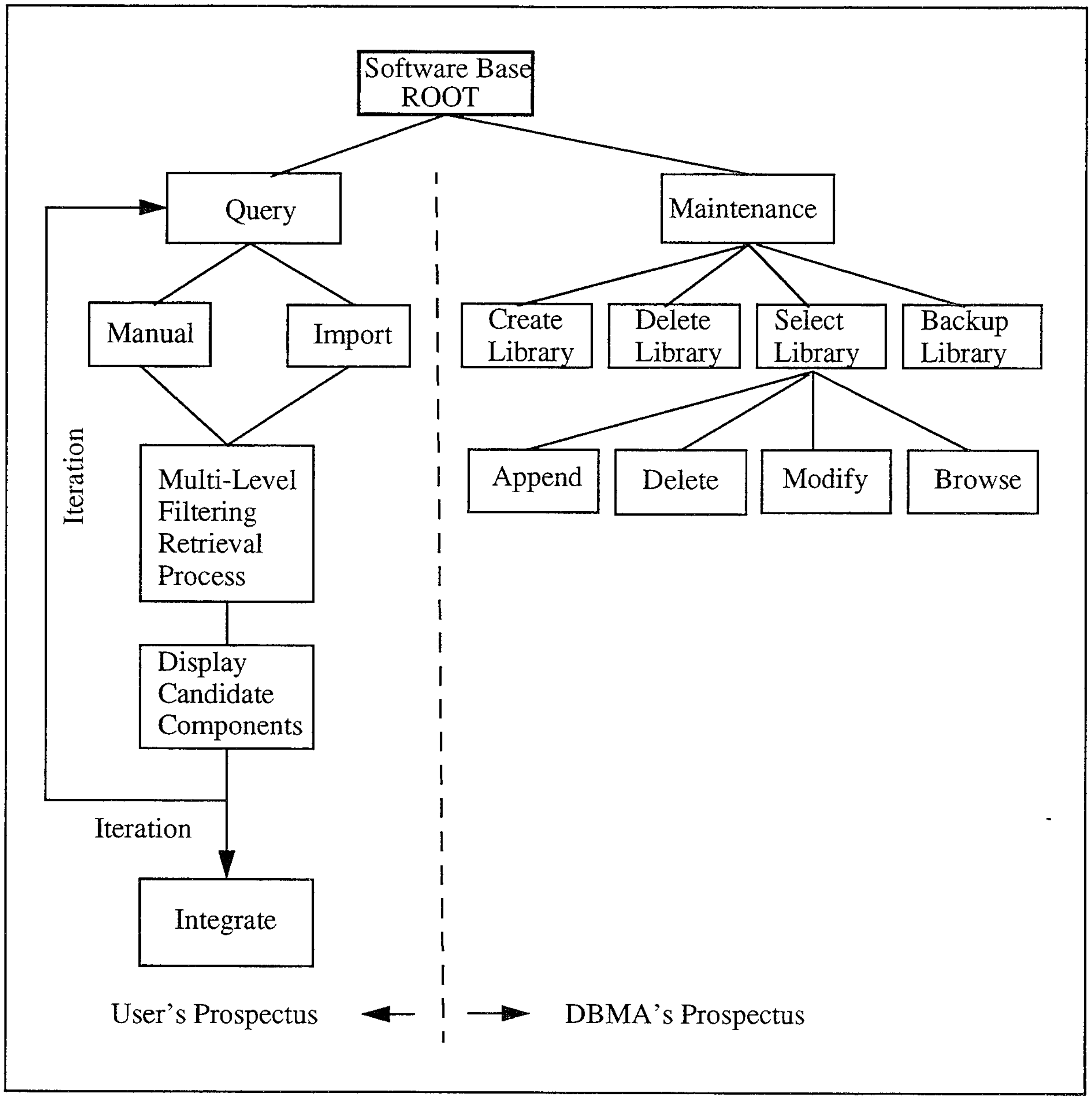


Figure 11. Software Base GUI Architecture Analysis

3. User Requirements

Although a SBGUI user could be any person from novice to an expert, we made some assumptions to narrow the range of user in order to get better utilization of the software base. These assumptions are:

- The user should be familiar with the basic operations of a GUI, such as mouse-driven, pull-down menu, drag and play, etc.

- The user should have some fundamental knowledge of the algebraic specification concept and at least one algebraic specification language, like OBJ3, etc
- The user should have a basic understanding of the concepts of reusable component retrieval techniques.
- The user should be familiar with the language which is used for the developing applications.

4. System Requirements

Since CAPS is an on-going project at the Naval Postgraduate School, the developed SBGUI should work under the UNIX based workstation such as SUN, Solborne, or other compatible workstation platforms. It should also run in the X-windows environment.

B. IMPLEMENTATION DETAIL

The final implementation includes 55 Ada programs, of which, 34 are codes generated in the TAE plus environment, a powerful user interface assistant tool. The remaining 21 programs are originally from the multi-level retrieval technique research in [Ref. 31]. This thesis has successfully integrated the user interface code for X-terminals and the MLFRT kernel code together to create a X-window based interactive software application. The software users manual is in Appendix D. The TAE Ada source code is in the Appendix E.

VI. CONCLUSION AND FUTURE RESEARCH

A. CONCLUSION

This thesis concentrates on the user interface and software database design for supporting the multi-level filtering retrieval technique. The user interface described has been implemented, yet it has not been integrated into CAPS. A key piece of the software database schema has been designed and validated, making follow-on work much easier. Also, a template module, which successfully addressed the language mismatch problem, has been designed. This eliminating the largest obstacle in implementing the software base. Figure 12 illustrates the kernels of the software base.

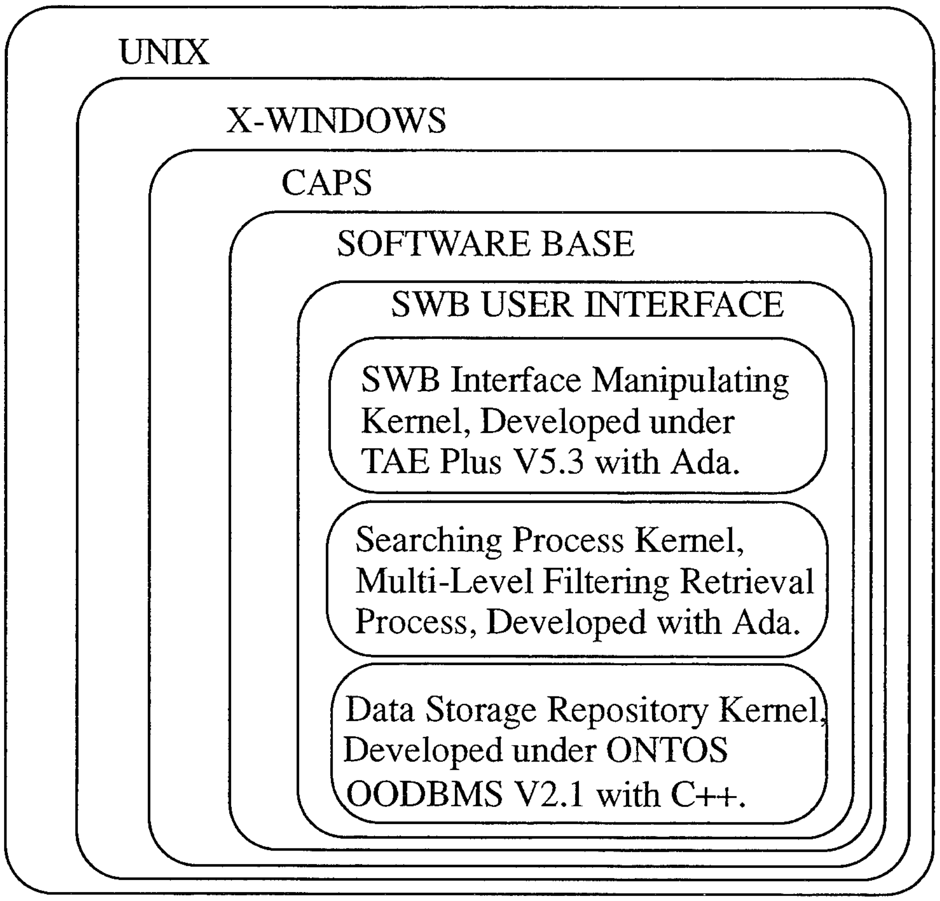

UNIX
X-WINDOWS
CAPS
SOFTWARE BASE
SWB USER INTERFACE
SWB Interface Manipulating Kernel, Developed under TAE Plus V5.3 with Ada.
Searching Process Kernel, Multi-Level Filtering Retrieval Process, Developed with Ada.
Data Storage Repository Kernel, Developed under ONTOS OODBMS V2.1 with C++.

Figure 12. Kernels of the Software Database

B. FUTURE RESEARCH

There are still many research areas following the work in this thesis. They are described as follows:

1. Increase Capability for More Types of Software Components

Currently the software base system contains only for the data structure components. Such a restriction should be relaxed in order to accommodate a variety of types of software components in the real world, such as communication components, window-manipulating metaphor components, etc. It should go further by covering the software architecture, requirements, documentation, users manual, etc.

2. Improve the Data Schema Design

This thesis accomplished a component schema design which meets the requirements of the MLFRT, and developed a working procedure for constructing and manipulating the physical data repository. However, a more extensive schema design is still needed, covering areas such as profiles, bag_of_profiles, etc. The database schema design should be an independent research topic in order to get complete design analysis and well-prepared documentation before going further.

3. Further Improvement for the Maintenance of the Database

This point is related to the previous concern. After the throughly researched data schema design, the implementation detail might be changed. The importance of maintaining the database will get more crucial as it grows. Because of this, the functionality of the maintenance work should be carefully reconsidered.

4. Populating the Database

According the convention of this thesis, there should be four attributes in each software component: Ada_Specification, Ada_Body, PSDL, and OBJ3. The Ada code currently exists, so the PSDL and OBJ3 need to be produced by some particular transformation process. [Ref. 32] has done the transformation for about 100 sets of the BOOCH Ada data structure components. However, the specification language OBJ3 is not intuitive for most software engineers. Populating the database effectively will require further research concerning automatic transformation of all varieties of types of the software components.

5. Testing

An extensive empirical study for the MLFRT is need. Under the situation with many data entries, the practical performance is unknown. [Ref. 31] performed a simple study to validate the MLFRT theory and compared its performance with the earlier software base research. It showed a high recall, high precision, and shorter search time but was based on a small mock up database. Further testing for a real consolidated database is needed.

6. Integrate the Retrieved Components

The most important target for the software base is to integrate the software component into the rapid prototyping system. Previous research about the software component integration was to contain all the relative information into a so-called Wrapper[Ref. 30] which serves as the bridge between the software base component and the PSDL specification. The MLFRT process computes the mapping from query operations and types to the corresponding operations and types used in the retrieved component. The integration process should be modified accordingly, to use this computed information instead of asking the user to supply it.

REFERENCE

1. Software Reuse Executive Primer, DOD Software Reuse Initiative Program Management Office, May 17, 1995.

2. Ted J. Biggerstaff and Charles Richter, “Reusability Framework, Assessment, and Directions”, IEEE Software, Vol 4, No. 2, pp. 41-49, March 1987.

3. Will Tracz, “A Conceptual Model for Metaprogramming”, Software Engineering Notes, Vol. 16, No. 3., ACM SIGSoft, pp. 36-45, July 1991.

4. Ruben Prieto-Diaz, “Making Software Reuse Work: An Implementation Model”, Software Engineering Notes, Vol. 16, No. 3., ACM SIGSoft, pp. 61-68, July 1991.

5. S. Cohen, Software Engineering Institute, “Process and Products for Software Reuse and Domain Analysis”, Symposium on Reuse and Re-engineering, National Institute of Software Quality and Productivity, Bethesda, MD., November 1991.

6. Reuse Report, Software Techniques Conference 95, November 1995.

7. Luqi and M. Ketabchi, “A Computer-Aided Prototyping System,” IEEE Transactions on Software Engineering, October 1988.

8. Luqi, Berzins, V. and Yeh, R., “A Prototyping Language for Real-Time Software,” IEEE Transactions on Software Engineering, 14, 10 (October, 1988), 1409-1423.

9. Dogan Ozdemir. The design and implementation of a reusable component library and a retrieval integration system, Master’s thesis, Naval Postgraduate School, Monterey, California, 1992.

10. Salton, G. and McGill, M., Introduction to Modern Information Retrieval, McGraw-Hill, 1983.

11. Luqi, Software Reuse without Reading Code, International Software Reuse Work Shop II, 1992.

12. Rubin Prieto-Diaz. Implementing faceted classification for software reuse. Communication of the ACM, pages 89-97, May 1991.

13. G. Fischer, Scott Henninger, and D. Redmiles. Cognitive tools for locating and comprehending software objects for rescue. In Proceedings, 13the International Conference on Software Engineering, May 1991.

14. E. Ostertag, J. Hendler, Rubin Prieto-Diaz, and C. Braun. Computing similarity in a reuse library system. ACM Transaction on Software Engineering and Methodology, pages 205-228, July 1992.

15. Scott Henninger. Using iterative refinement to find reusable software. IEEE Software, pages 48-59, September 1994.

16. Luqi, Proceedings of Software Engineering Workshop 94, Naval Postgraduate School, Monterey, California, 1994.

17. Luqi. Normalized specifications for identifying reusable software. In Proceedings of the 1987 Fall Joint Computer Conference, pages 46-49. IEEE, October 1987.

18. A. Mili, R. Mili, and R. Mittermeir. Storing and retrieving software components. In Proceedings 16th International Conference on Software Engineering, pages 15-19, 1994.

19. Eugene J. Rollins and Jeannette M. Wing. Specifications as search keys for software libraries. In Proceedings of the Eighth International Conference on Logic Programming, 1991.

20. Amy Moormann Zaremski and Jeannette M. Wing. Signature matching, a tool for using software libraries. In Proceedings, ACM Symposium on Foundations of Software Engineering, 1993. To appear, Transactions on Software Engineering and Methodology.

21. Amy Moormann Zaremski and Jeannette M. Wing. Specification matching of software components. Technical Report CMU-CS-95-127, School of Computer Science, Carnegie-Mellon University, 1995.

22. Neighbors, James M., The Draco Approach to Constructing Software from Reusable Components, IEEE Transactions on Software Engineering, V.10, pp. 564-574, September 1984.

23. Vogelsong, T. and Rothrock, J., Reusable Ada Products for Information Systems Development (RAPID) Lessons Learned During Pilot Operations, U.S. Army Information Systems Software Development Center - Washington, 1990.

24. Burton, Bruce A., Wienk, Rhonda, and others, The Reusable Software Library, IEEE Software, V. 4, pp. 25-33, July 1987.

25. Robert Steigerwald, Luqi, and John McDowell. CASE tool for reusable software component storage and retrieval in rapid prototyping. Information and Software Technology, pages 698-705, 1991.

26. Robert A. Steigerwald. Reusable Software Component Retrieval via Normalized Algebraic Specifications. Ph.D. thesis, Naval Postgraduate School, 1991.

27. Kokichi Futatsugi, Joseph Goguen, Jean-pierre Jouannaud, and Jose Meseguer. Principles of OBJ2. In Brian Reid, editor, Proceedings, Twelfth ACM Symposium on Principles of Programming Languages, pages 52-66. Association for Computing Machinery, 1985.

28. Joseph Goguen, Claude Kirchner, Helene Kirchner, Aristide Megrelis, and Jose Meseguer. An introduction to OJB3. In Jean-Pierre Jouannaud and Stephane Kaplan, editors, Prceed-

ings Conference on Conditional Term Rewriting, pages 258-263. Pringer, 1988. Lecture Notes inn Computer Science, Volume 308.

29. Joseph Goguen, Timothy Winkler, Jose Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In Joseph Goguen, editor, Algebraic Specification with OBJ: An Introduction with Case Studies. Cambridges, to appear, Also Technical Report, SRI International.

30. Scott J. Dolgoff. Automated interface for retrieving reusable software components, Master's thesis, Naval Postgraduate School, Monterey, California, 1992.

31. Doan Han Nguyen. An Architecture Model for Software Component Search. Ph.D. dissertation, Naval Postgraduate School, 1995.

32. Tuan Nguyen. Populating the software database for the Computer Aided Prototyping System (CAPS), Master's thesis, Naval Postgraduate School, Monterey, California, 1996.

33. John Kelly McDowell. A Reusable Component Retrieval System For Prototyping, Master's thesis, Naval Postgraduate School, Monterey, California, 1991.

34. Osman Ibrahim, Ph.D. Dissertation, "A Decision Support System for Software Requirements Engineering in a Rapid Prototyping Environment", (complete by Dec., 1996).

35. Ada 95 Rationale, Intermetrics Inc., Cambridge, Massachusetts, 1995.

36. Clayton Lewis, John Rieman, Task-Centered User Interface Design, - A Pratical Introduction, Shareware, Internet, 1994.

37. Joseph Goguen, Doan Nguyen, Jose Meseguer, Luqi, Du Zhang, Valdis Berzins, "Software Component Search", Journal of Systems Integration, Vol. 6, No 1-2, pages 93-134, 1996.

APPENDIX A - PARTIAL ORDERING AND HASSE DIAGRAM

A. PARTIAL ORDERING

We often use relations to order some or all of the elements of sets. For instance, we order words using the relation containing pairs of words (x, y) where x comes before y in the dictionary. We schedule projects using the relation consisting of pairs (x, y) where x and y are tasks in a project such that x must be completed before y begins. We order the set of integers using the relation containing the pairs (x, y) where x is less than y. When we add all of the pairs of the form (x, x) to these relations, we obtain a relation that is reflexive, anti-symmeric, and transitive. These are properties that characterize relations used to order the elements of sets using their relative size.

DEFINITION 1. A relation R on a set S is called a *partial ordering* or *partial order* if it is reflexive, antisymmetric, and transitive. A set S together with a partial ordering R is called a *partially ordered set*, or *poset*, and is denoted by (S, R).

In a poset the notation $a \leq b$ denotes that $(a, b) \in R$. This notation is used because the "less than or equal to" relation is a paradigm for a partial ordering. (Note that the symbol is used to denote the relation in any poset, not just the "less than or equals" relation.) The notation $a < b$ means that $a \leq b$, but $a \neq b$. Also, we say "a is less than b" or "b is greater than a" if $a < b$.

When a and b are elements of the poset $(S, \leq)$, it is not necessary that either $a \leq b$ or $b \leq a$. For instance, in *(P(Z),* $\subseteq$ *)*, {1, 2} is not related to {1, 3}, and vice versa, since neither set is contained within the other. Similarly, in $(Z, |)$, 2 is not related to 3 and 3 is not related to 2, since $2 \nmid 3$ and $3 \nmid 2$. This leads to the following definition.

DEFINITION 2. The elements a and b of a poset $(S, <)$ are called comparable if either $a \leq b$ or $b \leq a$. When a and b are elements of S such that neither $a \leq b$ nor $b \leq a$, a and b are called *incomparable.*

B. HASSE DIAGRAMS

Many edges in the directed graph for a finite poset do not have to be shown since they must be present. For instance, consider the directed graph for the partial ordering $\{ (a, b) | a \leq b \}$ on the set {1, 2, 3, 4}, shown in Figure 13(a). Since this relation is a partial ordering, it is reflexive, and its directed graph has loops at all vertices. Consequently, we do not have to show these loops since they must be present; in Figure 13(b) loops are not shown. Because a partial ordering is transitive, we do not have to show those edges that must be present because of transitivity.

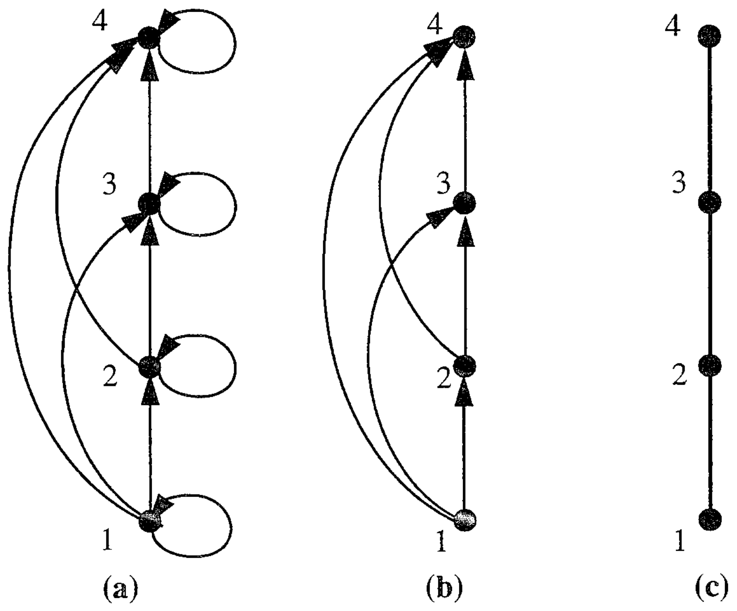


Figure 13. Constructing the Hasse Diagram for ({1, 2, 3, 4}, <=).

For example, in Figure 13(c) the edges (1, 3), (1, 4), and (2, 4) are not shown since they must be present. If we assume that all edges are pointed "upward" (as they are drawn in the figure), we do not have to show the directions of the edges; Figure 13(c) does not show directions.

In general, we can represent a partial ordering on a set using the following procedure. Start with the directed graph for this relation. Because a partial ordering is reflexive, a loop is present at every vertex. Remove these loops. Remove all edges that must be present because of the transitivity, since they must be present since a partial ordering is transitive. For instance, if (a, b) and (b, c) are in the partial ordering, remove the edge (a, c), since it must be present also. Furthermore, if (c, d) is also in the partial ordering, remove the edge(a,d), since is must be present also. Finally, arrange each edge so that its initial edge is below its terminal edge(as it is drawn on paper). Remove all the arrows on the directed edges, since all edges point "upward" toward their terminal vertex.

These steps are well-defined, and only a finite number of steps need to be carried out for a finite poset. When all the steps have been taken, the resulting diagram contains sufficient information to find the partial ordering. This diagram is called a Hasse Diagram, named after the 20th century German mathematician Helmut Hasse.

APPENDIX B - DATABASE DESIGN SOURCE CODE (ADA/C++)

A. Ada Source Code

```
-----------------------------------------------------------------------------
-
--
-- Component   : Component_PKG Spec
-- Author      : Ruey-Wen (Vincent) Hong
-- Date        : March 6, 1996
-- Language    : Ada
-- Compiler    : caps-suns7 SunAda
-- Description : The Ada spec mirror program for Ada/C++ Bridge Control
--               The Ada/C++ Bridge Control Technique is originally created
--               by Osman Ibrahim.
-- Purpose     : This package spec represents an Ada mirrored image for the
C++
--               class "Component". It encpsolates all operations and types
defined
--               over Component in a way making an Abstract Data Type of it.It
also
--               defines the interface between Ada operations defined over
--               Component and the coresponding C++ operations.
--               Procedure names are given followed by their interface
--               (C in all cases) and interface names. The a.ld pre-link
--               link_with pragmas are given in the file link_with_pragmas.a

--             Refer to this PKG body for some other detail regarding the
--             Conventions used for subprograms names

--             The method used to interface Ada and C++ code is as follow:

--             1- compile your C++ code you like to interface Ada to using a
--                C++ compiler

--             2- use the Unix nm command to get the symbolic name of the
--                subroutine you like to link your Ada to using
--                "pragma_interface" and pragme_interface_name. Choose the
--                symbolic name which is preceeded by "T". Use the symbolic
--                name in the pragma interface_name and the C++ subroutine
--                name in the pragma interface.

--             3- use link_with_pragma to pre-link with the C++ object code
--                in each of the files containing that code (.o) files. These
--                link with pragmas are included in the file
--                link_with_pragmas.a for this experimental application
--                template

-- Example    :  Suppose you have an Ada function called ada_new_Component and
--             you have a C++ function that implements the Ada function called
--             c_new_Component and the object code for the C++ function is
--             in the file Component_inteface.o and now you want to
interface the
--             Ada function to the coresponding C++ function :
--
--          1- Assuming Your C++ code is already compiled and you have the
(.o)
--             file(s).

--          2 use nm:
--                   >nm -ao Component_inteface.o | grep "c_new_Component"
--              - The output will be  a punch of names as follows:

-- Component_interface.o:000006c8 T _c_new_Component__FPciT2
-- Component_interface.o:00000000 - 00 0000  LSYM c_new_Component__FPciT2 :ZtF
-- Component_interface.o:000006c8 - 00 001c   FUN
c_new_Component__FPciT2:F(0,1

--             - Choose the first one (preceeded by "T"):
--                     "_c_new_Component__FPciT2" as the symbolic name.
--
--            Note : for details about why this symbolic name look this
--                   strange,  refer to a paper by Bjarne Stroustrup
--                   titled "Type-safe Linkage for C++"... by the way C++
--                   is one of his contributions.

--              - Now in your ada code, all you need is the following

--                Function ada_new_component(name      : in c_string;
--                                          level     : in integer ;
--                                          status    : in integer   )
--                                                        return component;

--             pragma interface(C,c_new_component);  -- C++ subroutine name
--             pragma interface_name(c_new_component,
"_c_new_component__FPciT2");

--               - This way you got an Ada function called ada_new_component
--                 so when you say somewhere in you Ada code:
--                 ada_new_component(some_name,some_integer,another_integer)
--                 you are actually calling the coresponding C++ function:
--                 c_new_component

--         3- Now for step #3 you have 3 ways to link with the C++ relevant
--            object files :
--                       - use withn  (not recommended by Ada documentations)
--                       - supply the C++ (.o) files  in the command
--                         line as opetions to a.ld , this is doable and
--                         can be included in the Makefile

--                        - The way I'm using is to use link_with pragma to
--                          link to the desired (.o) file

--                           ex: pragma link_with("Component_interface.o");
--
--               I've included all such pragma link_with in the file
--               link_with_pragmas.a and then my top level driver (interface)
--               includes a "with link_with_pragmas_PKG"
```

```
--
------------------------------------------------------------------------------
--

with a_strings;
use  a_strings;

with exception_interface_PKG;
use  exception_interface_PKG;

package Component_PKG is

   type Component_record is private;

   type Component is access Component_record;

   Function new_Component(name      : in a_string;
                          psdl      : in a_string;
                          ada_spec  : in a_string;
                          ada_body  : in a_string;
                          obj       : in a_string ) return Component;

   Function get_Component(name      : in a_string) return Component;

   Function GetComponentName(d      : in Component) return a_string;

   procedure SetComponentName(Name  : in a_string;
                              d     : in Component);

   procedure PutComponent(d     : in Component );

-----------------------------------------------------------------------

-- Component ITERATOR

-- The following operations lumps an iterator suitable for looping through
-- instances of the Component class and returning each of these instances
-- The following syntax of the Instance Iterator, although is given for
-- the Component Type, it is general enough to apply to any other TYPE under
-- conditition the Type MUST be classified into the DB (using Ontos CLASSIFY
-- utility) with the +X switch so that Ontos will maintain an aggregate of all
-- instance of that TYPE, in this case the Type is called "has an EXTENSION"
--  .. refer to ONTOS DB Tools and Utilities Guide Ch3.
-- WATCH OUT though that using +X swith has a perfomance degredation penality,
-- it slows down the application.

-- The syntax of using this iterator is as follows

--       Component_Iter := CREATE_ITERATOR ( type_name );

--       While Has_More_Elements(Component_Iter) loop

--       Component_Object := Get_Next_Element(Component_Iter)

--             do something with Component_Object
--             .....
--             ....

--       end loop
--           You can also issue:

--             RESET_ITERATOR(Component_Iter) : to re-iterate

--             DELETE_ITERATOR(Component_Iter) : to deallocate memory

--               Note also that Component_Iter and Component_Object are of the
same
--    type: Component of this PKG



   Function Create_Instance_Iterator(type_name : in a_string ) return
Component;

   Function Get_Next_Element(d : in Component ) return Component;

   procedure Reset_Iterator(d : in Component; type_name : in a_string );

   procedure Destroy_Iterator(d : in Component );

   Function Has_More_Elements(d : in Component ) return BOOLEAN;

----------------------------------------------------------------------

-- EXCEPTIONS

-- The 2 Exceptions "no_such_object" and "object_already_exists" applies to
-- all persistent types, that is why they are included in a separate PKG
-- "exception_interface_PKG", so they can be visible to all types.
-- Renaming both here is a matter of readability to know that we are
-- talking about Component Objects.

  no_such_Component        : exception renames no_such_object;

  Component_already_exists : exception renames object_already_exists;

------------------------------------------------------------------------

  private
         type Component_record is record
                 null;
         end record;

end Component_PKG;



------------------------------------------------------------------------------
-
--
-- Component    : Component_PKG Spec
-- Author       : Ruey-Wen (Vincent) Hong
-- Date         : March 6, 1996
-- Language     : Ada
-- Compiler     : caps-suns7 SunAda
-- Description : The Ada mirror program for Ada/C++ Bridge Control
-- Purpose     : This package Body represents an Ada mirrored image for the
C++
--               class "Component". It encpsolates all operations and types
defined
--               over Component in a way making an abstract DT of it.It also
--               defines the interface between Ada operations defined over
--               Component and the coresponding C++ operations.
--               Procedure names are given followed by their interface
--               (C in all cases) and interface name. The a.ld pre-link
--               link_with pragmas are also given.

--             Naming Conventions used for subprograms names are as follows:

--                1. Subprograms interfacing to coresponding C++ subprograms
--                   have the same name as the the coresponding C++
subprograms
--                   and these names are identified by the prefix "ada_c_"

--                2. The Ada subprograms implementing the functionality of the
--                   subprograms in 1. and do not include the interface detail
--                   like pragmas and c_strings are given the same name as the
--                   coresponding subprograms in 1. above without the prfix.

--             Refer to this PKG Spec for some other detail regarding the way
--             I used to interface Ada to a C++ code.
--
------------------------------------------------------------------------------
--

with a_strings;
use  a_strings;

with c_strings;
use  c_strings;

with db_utility_PKG;
use  db_utility_PKG;

with text_io;
use  text_io;

with exception_interface_PKG;
use  exception_interface_PKG;

package body Component_PKG is

------------------------------------------------------------------------------
-


   Function ada_c_new_Component(name      : in c_string;
                                psdl      : in c_string;
                                ada_spec  : in c_string;
                                ada_body  : in c_string;
                                obj       : in c_string ) return Component;

     pragma interface(C,ada_c_new_Component);
     pragma interface_name(ada_c_new_Component,
"_ada_c_new_Component__FPciT2");

-------------------------------------------------------------------------------
-

   Function new_Component(name      : in a_string;
                          psdl      : in a_string;
                          ada_spec  : in a_string;
                          ada_body  : in a_string;
                          obj       : in a_string ) return Component is

   C_name      : c_string := to_c(name);
   C_psdl      : c_string := to_c(psdl);
   C_ada_spec  : c_string := to_c(ada_spec);
   C_ada_body  : c_string := to_c(ada_body);
   C_obj       : c_string := to_c(obj);
   d           : Component;

   begin

       d := ada_c_new_Component(C_name, C_psdl, C_ada_spec, C_ada_body,
C_obj);

       case get_exception_code is

                    when normal_code                =>
                    null;
              when object_already_exists_code       =>
                      raise Component_already_exists;
              when others                           =>
                      raise ONTOS_Failure;
       end case;

       return d;

   end new_Component;

-------------------------------------------------------------------------------
-

   Function ada_c_find_Component(name      : in c_string) return Component;

     pragma interface(C,ada_c_find_Component);
     pragma interface_name(ada_c_find_Component,
"_ada_c_find_Component__FPc");

-------------------------------------------------------------------------------
-

   Function get_Component(name      : in a_string) return Component is

   C_name      : c_string := to_c(name);
   d           : Component;

   begin

       d := ada_c_find_Component(C_name);
```

```
      case get_exception_code is
              when normal_code                   =>
                     null;
              when no_such_object_code           =>
                     raise no_such_Component;
              when others                        =>
                     raise ONTOS_Failure;
      end case;

      return d;

  end get_Component;


-----------------------------------------------------------------------------
-
  Function ada_c_GetComponentName(d  : in Component) return c_string;

    pragma interface(C,ada_c_GetComponentName);
    pragma interface_name(ada_c_GetComponentName,
                           "_ada_c_GetComponentName__FP6Component");

-----------------------------------------------------------------------------
-


  Function GetComponentName(d  : in Component) return a_string is

  a_name       : a_string := to_a(ada_c_GetComponentName(d));

  begin

      case get_exception_code is
              when normal_code                   =>
                     null;
              when no_such_object_code           =>
                     raise no_such_Component;
              when others                        =>
                     raise ONTOS_Failure;
      end case;

      return a_name;

  end GetComponentName;


-----------------------------------------------------------------------------
-

  procedure ada_c_SetComponentName(
                  Name    : in c_string;
                  d       : in Component);

    pragma interface(C,ada_c_SetComponentName);
    pragma interface_name(ada_c_SetComponentName,
                           "_ada_c_SetComponentName__FPcP6Component");
-----------------------------------------------------------------------------
-
```

```
   procedure SetComponentName(Name     : in a_string;
                              d        : in Component   ) is

   C_name      : c_string := to_c(Name);

   begin

       ada_c_SetComponentName(C_name, d);

       case get_exception_code is
               when normal_code                    =>
                    null;
               when no_such_object_code            =>
                    raise no_such_Component;
               when others                         =>
                    raise ONTOS_Failure;
       end case;

   end SetComponentName;


------------------------------------------------------------------------------
-

   procedure PutComponent(d      : in Component) is

   -- The following 2 instantiations are for outputing
             -- expertise_level and status values respectively

             begin

         transaction_start;

         PUT("Component Name is ");
         PUT(GetComponentName(d).s);
         NEW_LINE;

         transaction_commit;

   end PutComponent;

------------------------------------------------------------------------------
-

-- Component ITERATOR

   Function ada_c_Create_Instance_Iterator(type_name : in c_string)
                         return Component;

     pragma interface(C,ada_c_Create_Instance_Iterator);
     pragma interface_name(ada_c_Create_Instance_Iterator,
                            "_ada_c_Create_Instance_Iterator__FPc");

------------------------------------------------------------------------------
-


   Function Create_Instance_Iterator(type_name : in a_string)
                                                                          return
```

```
Component is
                           begin
                                                          return
ada_c_Create_Instance_Iterator(to_c(type_name));

                           end Create_Instance_Iterator;

-----------------------------------------------------------------------------
-


   Function ada_c_Get_Next_Element(d : in Component) return Component;

     pragma interface(C,ada_c_Get_Next_Element);
     pragma interface_name(ada_c_Get_Next_Element,
                           "_ada_c_Get_Next_Element__FP16InstanceIterator");

-----------------------------------------------------------------------------
-


   Function Get_Next_Element(d : in Component) return Component is

                           begin
                                                          return
ada_c_Get_Next_Element(d);

                           end Get_Next_Element;

-----------------------------------------------------------------------------
-


   procedure ada_c_Reset_Iterator(d : in Component; type_name : in c_string);

     pragma interface(C,ada_c_Reset_Iterator);
     pragma interface_name(ada_c_Reset_Iterator,
                           "_ada_c_Reset_Iterator__FP16InstanceIteratorPc");

-----------------------------------------------------------------------------
-


   procedure Reset_Iterator(d : in Component; type_name : in a_string) is

                           begin
                                                          ada_c_Reset_Iterator(d,
to_c(type_name));

                           end Reset_Iterator;


-----------------------------------------------------------------------------
-


   procedure ada_c_Destroy_Iterator(d : in Component);

     pragma interface(C, ada_c_Destroy_Iterator);
     pragma interface_name( ada_c_Destroy_Iterator,
                         "_ada_c_Destroy_Iterator__FP16InstanceIterator");

-------------------------------------------------------------------------------
-


   procedure Destroy_Iterator(d : in Component) is

                           begin

                                                     ada_c_Destroy_Iterator(d);

                           end Destroy_Iterator;


-------------------------------------------------------------------------------
-


   Function ada_c_Has_More_Elements(d : in Component) return BOOLEAN;

     pragma interface(C,ada_c_Has_More_Elements);
     pragma interface_name(ada_c_Has_More_Elements,
                         "_ada_c_Has_More_Elements__FP16InstanceIterator");

   Function Has_More_Elements(d : in Component) return BOOLEAN is

                           begin

                                                     return
ada_c_Has_More_Elements(d);

                           end Has_More_Elements;


-------------------------------------------------------------------------------
-

end Component_PKG;

-------------------------------------------------------------------------------
-- Component   : Component_ops.a
-- Author      : Ruey-Wen (Vincent) Hong
-- Date        : March 6, 1996
-- Language    : Ada
-- Compiler    : caps-suns7 SunAda

-- Description : The Ada mirror program for Ada/C++ Bridge Control
--               The Ada/C++ Bridge Control Technique is originally created
--               by Osman Ibrahim.

--               This module has originally appeared in ECS under the same
--               name (Component_ops.a) and was coded completely in C++
--               I translated it into Ada to examine the possibilty of
--               directly using C++ classes from inside ADA and thus
--               testing the new approach. It should be noted that without
--               the new approach of interfacing Ada to C++, this module
--               could not be coded in Ada
--               This Ada module provide the same functionality that is
--              currently provided by the coresponding C++ module for
--              the Component pool in the ECS.
--              I did not try to change any logic or implementation here
--              to test the new approach.

--                  Notice that some of the code here is redundant and is not
--                  needed especialy the checks to see if the Component
already
--                  exists in the DB (or the parallel check to see if the
--                  Component does not exists in the DB) before performing
some
--                  operations. The lower level operations defined in the
--                  Component and db_utility packages guards aginst the
--                  occurences of such conditions by raising the proper
--                  exception.

------------------------------------------------------------------------------
-

with link_with_pragmas_PKG;
use link_with_pragmas_PKG;

with Component_PKG;
use Component_PKG;

with db_utility_PKG;
use  db_utility_PKG;

with exception_interface_PKG;
use  exception_interface_PKG;

with c_strings;
use c_strings;

with a_strings;
use a_strings;

with u_env;
use u_env;

with text_io;
use text_io;

------------------------------------------------------------------------------
-


package Component_Ops_PKG is


 package int_io is new integer_io(integer);
 use  int_io;

  procedure create_Component (name      : in a_string;
                              psdl      : in a_string;
                              ada_spec  : in a_string;
                              ada_body  : in a_string;
                              obj       : in a_string );
  procedure write_Components_to_file;

  procedure add_Component (name      : in a_string;
                           psdl      : in a_string;
                           ada_spec  : in a_string;
                           ada_body  : in a_string;
                           obj       : in a_string );


  procedure delete_Component (name : in a_string);

  procedure show_Component(name : in a_string);

  procedure show_all_Components;

end Component_Ops_PKG;

------------------------------------------------------------------------------
-

package body Component_Ops_PKG is

------------------------------------------------------------------------------
-

-- This procedure iterates through the Component instances (using the new
-- iterator) and write Component info into a file called:
-- "/.caps/temp/ddbdisplay" to be used later by TAE to display this info
-- in the Component panel. Notice the use of the new ITERATOR here.


  procedure write_Components_to_file is

  local_Component     : Component;
  Component_iterator  : Component;

   user_directory: a_string :=
         copy(c_strings.to_a(u_env.getenv(c_strings.to_c("HOME"))) &
         a_strings.to_a("/.caps/temp/ddbdisplay")) ;
   ECS_output : file_type;
   ECS_output_file_name : a_strings.a_string := user_directory;
   begin

        transaction_start;

        open(ECS_output, MODE => OUT_FILE, NAME => ECS_output_file_name.s);

        Component_iterator := Create_Instance_Iterator(to_a("Component"));

        while ( Has_More_Elements(Component_iterator)) loop

             local_Component := Get_Next_Element(Component_iterator);

             put(ECS_output,GetComponentName(local_Component).s);

             NEW_LINE(ECS_output);
         end loop;
         close(ECS_output);
         transaction_commit;
```

```
     end write_Components_to_file;

-----------------------------------------------------------------------------
-


  procedure create_Component (name      : in a_string;
                              psdl      : in a_string;
                              ada_spec  : in a_string;
                              ada_body  : in a_string;
                              obj       : in a_string  ) is

  local_Component : Component;

   begin

     local_Component := new_Component(name, psdl, ada_spec, ada_body, obj);

    end create_Component;

-----------------------------------------------------------------------------
-

-- Add Component to the DB
-- Note that when adding a new Component to the DB his status is free by
default
-- that is the reason we do not need status as a parameter.

  procedure add_Component (name      : in a_string;
                           psdl      : in a_string;
                           ada_spec  : in a_string;
                           ada_body  : in a_string;
                           obj       : in a_string ) is

  local_Component : Component;

  begin

      transaction_start;
      local_Component := get_Component(name);
      if local_Component /= null then
         PUT_LINE("Component already exists in the DB");
         null;
      else
         local_Component:= new_Component(name, psdl, ada_spec, ada_body, obj);
         save_to_db(local_Component);
      end if ;
      transaction_commit;

   end add_Component;

-----------------------------------------------------------------------------
-


  procedure delete_Component (name : in a_string) is

  local_Component : Component;
```

```
   begin

      transaction_start;
      local_Component := get_Component(name);
      if local_Component = null then
         PUT_LINE("Component does not exist in the DB");
       null;
      else
         delete_from_db(local_Component);
      end if ;
      transaction_commit;

   end delete_Component;

-------------------------------------------------------------------------------
-

procedure show_Component(name: in a_string) is

  local_Component : Component;

   begin
      transaction_start;
      local_Component := get_Component(name);
      if local_Component = null then
         PUT_LINE("Component does not exist in the DB");
       null;
      else
         PutComponent(local_Component);
      end if ;
      transaction_commit;

   end show_Component;

-------------------------------------------------------------------------------
-

procedure show_all_Components is

 local_Component     : Component;
 Component_iterator  : Component;

  begin

    transaction_start;
    Component_iterator := Create_Instance_Iterator(to_a("Component")) ;
    while ( Has_More_Elements(Component_iterator)) loop
    local_Component := Get_Next_Element(Component_iterator);
    if local_Component = null then
         PUT_LINE("Component does not exist in the DB");
      null;
     else
         PutComponent(local_Component);
      end if ;

  end loop;
  transaction_commit;
  end show_all_Components;

-----------------------------------------------------------------------------
-

end Component_Ops_PKG;

-----------------------------------------------------------------------------
-
-----------------------------------------------------------------------------
-
--
-- Component   : Db_Utility_PKG Spec
-- Author      : Ruey-Wen (Vincent) Hong
-- Date        : March 6, 1996
-- Language    : Ada
-- Compiler    : caps-suns7 SunAda
-- Description : The Ada mirror program for Ada/C++ Bridge Control
--               The Ada/C++ Bridge Control Technique is originally created
--               by Osman Ibrahim.
-- Purpose     : This package spec represents an Ada mirrored image for some
of
--               Ontos Free functions that access and manipulate objects and
--               other DB operations. These operations needs to be extended in
--               the future in the same way as the need arises.
-----------------------------------------------------------------------------
--

with a_strings;
use a_strings;

with Component_PKG;
use  Component_PKG;

package db_utility_PKG is

-- General ONTOS operations

   procedure open_database(ddb : in a_string);

   procedure close_database(ddb : in a_string);

   procedure transaction_start;

   procedure transaction_commit;

   procedure save_to_db(d   : in Component);

   procedure delete_from_db(d : in Component);

-----------------------------------------------------------------------------

end db_utility_PKG;



-----------------------------------------------------------------------------
-
```

```
--
-- Component   : Db_Utility_PKG Body
-- Author      : Ruey-Wen (Vincent) Hong
-- Date        : March 6, 1996
-- Language    : Ada
-- Compiler    : caps-suns7 SunAda
-- Description : The Ada mirror program for Ada/C++ Bridge Control
--               The Ada/C++ Bridge Control Technique is originally created
--               by Osman Ibrahim.
-- Purpose:    This package body represents an Ada mirrored image for some of
--             Ontos Free functions that access and manipulate objects and
--             other DB operations. These operations needs to be extended in
--                                                            the future in the same
way as the need arises.
--             Procedure names are given followed by their interface
--             (C in all cases) and interface name.
--
--             Refer to the Component_PKG body for some other detail regarding
--             the Conventions used for subprograms names

--             Refer to the Component PKG Spec for some other detail regarding
--             the way I used to interface Ada to a C++ code.

--             Refer to the file exception_interface.h for the meaning of each
--             exceptiuons used here.
--
-------------------------------------------------------------------------------
--

with a_strings;
use a_strings;

with c_strings;
use c_strings;

with exception_interface_PKG;
use  exception_interface_PKG;

with Component_PKG; -- needed for type "Component" to be visible here which
I do
                   -- not think it is right, it is needed because some
functions
                   -- have "Component" as an input parameter..refer to the
note
                                                                              --
below.
use  Component_PKG;

package body db_utility_PKG is

-------------------------------------------------------------------------------
--

   procedure ada_c_open_database(ddb : in c_string) ;

     pragma interface(C,ada_c_open_database);
     pragma interface_name(ada_c_open_database, "_ada_c_open_database__FPc");

-------------------------------------------------------------------------------
```

```
--

   procedure open_database(ddb : in a_string) is

                           begin

                                          ada_c_open_database(to_c(ddb));

                     case get_exception_code is
                when normal_code                 =>
                       null;
                when db_open_failed_code         =>
                       raise db_open_failed;
                when others                      =>
                       raise Ontos_failure;
                     end case;


            end open_database;

-------------------------------------------------------------------------------
-

   procedure ada_c_close_database(ddb : in c_string);

     pragma interface(C,ada_c_close_database);
     pragma interface_name(ada_c_close_database,
"_ada_c_close_database__FPc");

-------------------------------------------------------------------------------
-


   procedure close_database(ddb : in a_string) is

      begin

          ada_c_close_database(to_c(ddb));

          case get_exception_code is
               when normal_code                 =>
                      null;
               when db_not_open_code            =>
                      raise db_not_open;
               when others                      =>
                      raise Ontos_failure;
          end case;

      end close_database;

-------------------------------------------------------------------------------
-


   procedure ada_c_transaction_start;

     pragma interface(C,ada_c_transaction_start);
     pragma interface_name(ada_c_transaction_start,
                           "_ada_c_transaction_start__Fv");
```

```
------------------------------------------------------------------------------
-

   procedure transaction_start is

                           begin

                                                 ada_c_transaction_start;

         case get_exception_code is
               when normal_code                   =>
                      null;
               when others                        =>
                      raise Ontos_failure;
          end case;

                           end transaction_start;

------------------------------------------------------------------------------
-

   procedure ada_c_transaction_commit;

     pragma interface(C,ada_c_transaction_commit);
     pragma interface_name(ada_c_transaction_commit,
                           "_ada_c_transaction_commit__Fv");

   procedure transaction_commit is

                           begin

                                                               ada_c_transaction_commit;

                case get_exception_code is
               when normal_code                   =>
                      null;
               when no_active_transaction_code    =>
                      raise no_active_transaction;
               when others                        =>
                      raise Ontos_failure;
                end case;


                           end transaction_commit;

------------------------------------------------------------------------------
-

-- I think the following 2 operations should be moved to the Component ADT and
-- renamed "save_Component__to_db" and "delete_Component_from_db" respectively
-- because they depend on the type of object passed and we can not make the
-- input type generic ... can we?????

------------------------------------------------------------------------------
-

   procedure ada_c_save_to_db(d  : in Component);
```

```
     pragma interface(C,ada_c_save_to_db);
     pragma interface_name(ada_c_save_to_db,
"_ada_c_save_to_db__FP6Component");

-----------------------------------------------------------------------------
-


   procedure save_to_db(d  : in Component) is

                         begin

                                                    ada_c_save_to_db(d);

            case get_exception_code is

               when normal_code                   =>
                      null;
               when object_already_exists_code    =>
                      raise object_already_exists;
               when others                        =>
                      raise Ontos_failure;
           end case;


                         end save_to_db;
-----------------------------------------------------------------------------
-

   procedure ada_c_delete_from_db(d : in Component);

     pragma interface(C,ada_c_delete_from_db);
     pragma interface_name(ada_c_delete_from_db,
                         "_ada_c_delete_from_db__FP6Component");

-----------------------------------------------------------------------------
-


   procedure delete_from_db(d : in Component) is

                         begin

                                                    ada_c_delete_from_db(d);

            case get_exception_code is

               when normal_code                   =>
                      null;
               when no_such_object_code           =>
                      raise no_such_object;
               when others                        =>
                      raise Ontos_failure;
           end case;


                         end delete_from_db;
```

```
-----------------------------------------------------------------------

end db_utility_PKG;

------------------------------------------------------------------------------
-



------------------------------------------------------------------------------
-
--
-- Component   : exception_PKG Spec
-- Author      : Ruey-Wen (Vincent) Hong
-- Date        : March 6, 1996
-- Language    : Ada
-- Compiler    : caps-suns7 SunAda
-- Description : The Ada spec mirror program for Ada/C++ Bridge Control
--               The Ada/C++ Bridge Control Technique is originally created
--               by Osman Ibrahim.
-- Purpose:     This package spec represents an Ada mirrored image for the
--              coresponding C++ unit "exception_interface". It is made in a
--              separate PKG because most of these exceptions apply to all
--              types. In the future it will be easy to use those exceptions
--              as is or renamed to suit a specific type.

--              One function from the coresponding C++ unit "exception_
--              interface" is missing intentionally here which is :
--              "set_exception_code()" because it is not needed to be visible
--              in the Ada side.

--              Refer to the unit "exception_interface.h" for the meaning
--              of each of the exceptions defined here.
------------------------------------------------------------------------------
--

package exception_interface_PKG is

------------------------------------------------------------------------------

-- EXCEPTIONS

   Ontos_failure            :  exception;

   db_open_failed           :  exception;

   db_not_open              :  exception;

   no_active_transaction    :  exception;

   no_such_object           :  exception;

   object_already_exists    :  exception;

---------------------------------------------------------------------------

-- Exception codes captured and returned to ada from ONTOS
-- Refer to the comment in the PKG body.

   type exception_code is (normal_code,
                                object_already_exists_code,
                                no_such_object_code,
                                db_open_failed_code,
                                db_not_open_code,
                                no_active_transaction_code,
                                Ontos_failure_code          );

-----------------------------------------------------------------------------
-


   Function get_exception_code return exception_code;

-----------------------------------------------------------------------------

end exception_interface_PKG;



-----------------------------------------------------------------------------
-
--
-- Component     : exception_PKG Body
-- Author        : Ruey-Wen (Vincent) Hong
-- Date          : March 6, 1996
-- Language      : Ada
-- Compiler      : caps-suns7 SunAda
-- Description : The Ada spec mirror program for Ada/C++ Bridge Control
--                 The Ada/C++ Bridge Control Technique is originally created
--                 by Osman Ibrahim.
-- Purpose:      This package body represents an Ada mirrored image for the
--               coresponding C++ unit "exception_interface". It is made in a
--               separate PKG because most of these exceptions apply to all
--               types. In the future it will be easy to use those exceptions
--               as is or renamed to suit a specific type.

--               One function from the coresponding C++ unit "exception_
--               interface" is missing intentionally here which is :
--               "set_exception_code()" because it is not needed to be visible
--               in the Ada side.

--               Refer to the unit "exception_interface.h" for the meaning
--               of each of the exceptions defined here.
-----------------------------------------------------------------------------
--

with text_io;
use  text_io;

package body exception_interface_PKG is

-----------------------------------------------------------------------------
--

-- This function was introduced to allow ada to capture an exception raised
-- inside ONTOS so that Ada can handle it in a way taht will not cause
-- the program to abort because of a user error; e.g a misspelled DB name.

-----------------------------------------------------------------------------
```

```
--

   Function ada_c_get_exception_code return integer;

     pragma interface(C,ada_c_get_exception_code);
     pragma interface_name(ada_c_get_exception_code,
                           "_ada_c_get_exception_code__Fv");

------------------------------------------------------------------------------
--

   Function get_exception_code return exception_code is


    begin

      case ada_c_get_exception_code is

            when exception_code'POS(normal_code)                        =>
                  return normal_code;

            when exception_code'POS(no_such_object_code)                =>
                  return no_such_object_code;

            when exception_code'POS(object_already_exists_code)         =>
                  return object_already_exists_code;

            when exception_code'POS(db_open_failed_code)                =>
                  return db_open_failed_code;

            when exception_code'POS(db_not_open_code)                   =>
                  return object_already_exists_code;

            when exception_code'POS(no_active_transaction_code)         =>
                  return no_active_transaction_code;

            when others                                                 =>
                  return Ontos_failure_code;

      end case;

    end get_exception_code;

------------------------------------------------------------------------------
--

end exception_interface_PKG;


------------------------------------------------------------------------------
-
--
-- Component   : link_with_pragmas_PKG Spec
-- Author      : Ruey-Wen (Vincent) Hong
-- Date        : March 6, 1996
-- Language    : Ada
-- Compiler    : caps-suns7 SunAda
-- Description : The Ada spec mirror program for Ada/C++ Bridge Control
--                The Ada/C++ Bridge Control Technique is originally created
--                by Osman Ibrahim.
-- Purpose:     This package the pragma link_with for the a.ld pre-link to the
--              the relevant C++ components given by thier object files.
--              It also contains pragma link_with for the a.ld pre-link to the
--              procedure cplusplus_init which is required at TAE level to make
--              TAE, Ada, C++, and ONTOS behave friendlt together !!!!!!
--
--              For linking with a C++ (generally foreign lang.) object file,
--              there are 3 ways to do that :

--                          - use withn  (not recommended by Ada
documentations)

--                          - supply the C++ (.o) files  in the command
--                            line as opetions to a.ld , this is doable and
--                             can be included in the Makefile

--                          - The way I'm using is to use link_with pragma to
--                             link to the desired (.o) file

--              Refer to the files designer_s.a and designer_b.a for other
--              relevant details of how interfacing Ada to C++

-------------------------------------------------------------------------------
--

package link_with_pragmas_PKG is

-- CAPS C++ operation to initialize static constructors

   procedure cplusplus_init;
     pragma interface(C,cplusplus_init);
     pragma interface_name(cplusplus_init, "_main");

-------------------------------------------------------------------------------
-

-- Main C++ interface object module, required for initialization of C++
-- static constructors.

   pragma link_with("C++_initialize.o");

   pragma link_with("component.o");

   pragma link_with("component_interface.o");

   pragma link_with("db_utility.o");

   pragma link_with("exception_interface.o");

end link_with_pragmas_PKG;
```

B. C++ Source Code

```
/* --------------------------------------------------------------------------

-- Unit            : Class Component (.h)
-- File            : component.h
-- Date            : Documented Feb 29, 1996.
-- Author          : Ruey-Wen (Vincent) Hong
-- Systems         : Sun C++ and ONTOS (2.1)
-- Description     : The Ada/C++ Bridge Control Technique is originally created
--                 : by Osman Ibrahim.

--                 : The header for the Component Class that implement
--                 : the Software Component ADT in C++

----------------------------------------------------------------------------
*/


#include <Object.h>

class Component : public Object
{
  private:

    char* priv_psdl;     // The component's psdl spec
    char* priv_ada_spec; // The component's ada spec
    char* priv_ada_body; // The component's ada body
    char* priv_obj;      // The component's OBJ code

  public:

    // Constructors

    Component(char* name=(char*)0,
              char* psdl=(char*)0,
              char* ada_soec=(char*)0,
              char* ada_body=(char*)0,
              char* obj=(char*)0 );

    Component (APL*);      // (Ontos required Constructor)

    // Ontos required member function)

    Type *getDirectType();

    // Accessors

    char*   GetComponentPsdl() ;

    void    SetComponentPsdl(char* psdl);

};

/* --------------------------------------------------------------------------

-- Unit            : class Component implementation (.cxx)
-- File            : Componentc.cxx
-- Date            : Documented Feb 29, 1996.
-- Author          : Ruey-Wen (Vincent) Hong
-- Systems          : Sun C++ and ONTOS (2.1)
-- Description      : The Ada/C++ Bridge Control Technique is originally created
--                  : by Osman Ibrahim.
--                  : Provides the implementation (definition) for the Component
                      Class that implements the Component ADT in C++
------------------------------------------------------------------------
*/

#include "component.h"
#include <Directory.h>

//----------------------------------------------
// constructors
//----------------------------------------------

/* ---------------------------------------------------------------------
*/

Component::Component(APL *theAPL) : Object(theAPL)
{
}

/* ---------------------------------------------------------------------
*/

Component::Component(char* name,
                     char* psdl,
                     char* ada_spec,
                     char* ada_body,
                     char* obj ): Object(name)

{
  initDirectType((Type *)OC_lookup("Component"));

  priv_psdl     = psdl;
  priv_ada_spec = ada_spec;
  priv_ada_body = ada_body;
  priv_obj      = obj;
}

//----------------------------------------------
// accessors
//----------------------------------------------

Type *Component::getDirectType()
{
  return (Type*)OC_lookup("Component");
}


/* ---------------------------------------------------------------------
*/

void Component::SetComponentPsdl(char* psdl)
{
   priv_psdl = psdl;
}

/* ---------------------------------------------------------------------
```

```
*/

char* Component::GetComponentPsdl()
{
  return priv_psdl;
}

/* -------------------------------------------------------------------------
*/



/* --------------------------------------------------------------------------
-- Unit            : Component_Interface (.h)
-- File            : component_interface.h
-- Date            : Documented Feb 29, 1996.
-- Author          : Ruey-Wen (Vincent) Hong
-- Systems         : Sun C++ and ONTOS (2.1)
-- Description     : The Ada/C++ Bridge Control Technique is originally created
--                 : by Osman Ibrahim.

--                 :  The sole reason for this unit is to allow ADA to Create,
                      Access, and manipulate objects (instances) of the
                      "Component Class". We tried to do that directly without
that
                      second level interface, but we did not succeed.
                      Comments about how ada can communicate with code written
                      in C++ can be found in some of the Ada files in this Dir.
-------------------------------------------------------------------------
*/

#include <Database.h>
#include <Directory.h>
#include "component.h"
#include <Type.h>


// The following operation is just for making ada able to call the
constructors
// of the Component Class.

Component*  ada_c_new_component( char* Myname,
                                 char* MyPsdl,
                                 char* MyAdaSpec,
                                 char* MyAdaBody,
                                 char* MyObj );

// The following operation is to allow Ada to lookup and retrieve an instance
// of the Component Class

Component*  ada_c_find_component(char*);

// The following Operations each coresponds to a member function of the
Component
// Class.

char*    ada_c_GetComponentName(Component* ada_ptr);

void     ada_c_SetComponentName(char* name, Component* ada_ptr);
```

```
//---------------- component ITERATOR---------------------

// The following operations lumps an iterator suitable for looping through
// instances of the Component class and returning each of these instances
// The following syntax of the Instance Iterator, although is given for
// the Component Type, it is general enough to apply to any other TYPE under
// conditition the Type MUST be classified into the DB (using Ontos CLASSIFY
// utility) with the +X switch so that Ontos will maintain an aggregate of all
// instance of that TYPE, in this case the Type is called "has an EXTENSION"
//  .. refer to ONTOS DB Tools and Utilities Guide Ch3.
// WATCH OUT though that using +X swith has a perfomance degredation penality,
// it slows down the application.

InstanceIterator* ada_c_Create_Instance_Iterator(char* type_name);

Component*           ada_c_Get_Next_Element(InstanceIterator* it);

void                ada_c_Reset_Iterator(InstanceIterator* it, char* type_name);

void                ada_c_Destroy_Iterator(InstanceIterator* it);

OC_Boolean          ada_c_Has_More_Elements(InstanceIterator* it);


/* -----------------------------------------------------------------------------

-- Unit             : The implementation for Component_interface (.cxx)
-- File             : Component_interface.cxx
-- Date             : Documented Feb 29, 1996.
-- Author           : Ruey-Wen (Vincent) Hong
-- Systems          : Sun C++ and ONTOS (2.1)
-- Description      : The Ada/C++ Bridge Control Technique is originally created
--                  : by Osman Ibrahim.
--                  : The sole reason for this unit is to allow ADA to Create,
                      Access, and manipulate objects (instances) of the
                      "Component Class". We tried to do that directly without
that
                      second level interface, but we did not succeed.
                      Comments about how ada can communicate with code written
                      C++ can be found in some of the Ada files in this Dir.
----------------------------------------------------------------------------
*/

#include "component_interface.h"
#include <Exception.h>
#include "exception_interface.h"


// Note : We tried to place all EXCEPTION objects in one header file
(exception
// _interface.h) and be used whereever needed, but this resulted in an error
// came from the loader saying they are multiply defined, the same error came
// out when even we placed them global in the same file, so we had to put each
// in the proper function where the Exception is expected.

//==========================================================================
// The following operation is just for making ada able to call the
constructors


Component* ada_c_new_Component( char* Myname,
                                char* MyPsdl,
                                char* MyAdaSpec,
                                char* MyAdaBody,
                                char* MyObj )
{

  ExceptionHandler  object_already_exists("NameInUse");

  if (object_already_exists.Occurs())
               ada_c_set_exception_code(object_already_exists_code);
  else
      {
  ada_c_set_exception_code( normal_code);

  Component* aComponent =  new Component(Myname,
MyPsdl,
                                         MyAdaSpec,
MyAdaBody,
                                         MyObj);

  return aComponent;
      }
}

//==========================================================================

Component* ada_c_find_Component(char* Component_name)

{

  ExceptionHandler  no_such_object ("NameNotFound");

  if (no_such_object.Occurs())
              ada_c_set_exception_code(no_such_object_code);
  else
      {
 ada_c_set_exception_code( normal_code);

 Component *aComponent =
(Component*)OC_lookup(Component_name);

    return aComponent;
      }

}

//==========================================================================

char*  ada_c_GetComponentName(Component* ada_ptr)

{
  ExceptionHandler  no_such_object ("NameNotFound");

  if (no_such_object.Occurs())
```

```
                ada_c_set_exception_code(no_such_object_code);
  else
      {
        ada_c_set_exception_code( normal_code);

 return ada_ptr->Name();
      }
}

//==========================================================================

void  ada_c_SetComponentName(char* name, Component* ada_ptr)

{
  ExceptionHandler  no_such_object ("NameNotFound");

  if (no_such_object.Occurs())
                ada_c_set_exception_code(no_such_object_code);
  else
      {
        ada_c_set_exception_code( normal_code);

 ada_ptr->Name(name);
      }
}

//==========================================================================

//------------------ Component ITERATOR---------------------

 InstanceIterator* ada_c_Create_Instance_Iterator(char* type_name)

{
   InstanceIterator* it = new  InstanceIterator((Type*) OC_lookup(type_name));
   return it;
}

Component* ada_c_Get_Next_Element(InstanceIterator* it)

{
  Component* next_Component = (Component*) (Entity*)it->operator()();
  return next_Component;
}

void ada_c_Reset_Iterator(InstanceIterator* it, char* type_name)

{
     it->Reset((Type*) OC_lookup(type_name));
}

void ada_c_Destroy_Iterator(InstanceIterator* it)

{
     it->Destroy();
}

OC_Boolean ada_c_Has_More_Elements(InstanceIterator* it)

{
  return it->moreData();
}

//=========================================================================



/* ---------------------------------------------------------------------------

-- Unit            : Header for db_utility (.h)
-- File            : db_utility.h
-- Date            : Documented Feb 29, 1996.
-- Author          : Ruey-Wen (Vincent) Hong
-- Systems         : Sun C++ and ONTOS (2.1)
-- Description     : The Ada/C++ Bridge Control Technique is originally created
--                 : by Osman Ibrahim.
--                 :  Provide Function prototypes of some relevant Ontos DB
                      operation so that they are visible to and callable from
                      Ada. These operations needs to be extended in the future
                      in the same way as the need arises.

---------------------------------------------------------------------------
*/

#include <Database.h>
#include "component.h"
#include "exception_interface.h"

int    ada_c_open_database(char* dbname);

void   ada_c_close_database(char* dbname);

void   ada_c_transaction_start();

void   ada_c_transaction_commit();

void   ada_c_save_to_db(Component*);

void   ada_c_delete_from_db(Component*);

/* ---------------------------------------------------------------------------

-- Unit            :  The implementation for db_utility (.cxx)
-- File            :  db_utility.cxx
-- Date            :  Documented Oct 5,1995.
-- Author          :  Osman Ibrahim
-- Systems         :  Sun C++ and ONTOS (2.1)
-- Description     :  Provide the implementation for some relevant Ontos DB
                      operation so that they are visible to and callable from
                      Ada. These operations needs to be extended in the future
                      in the same way as the need arises.

---------------------------------------------------------------------------
*/

#include "component_interface.h"
#include <Directory.h>
#include <Exception.h>
#include "exception_interface.h"
```

```
// Note : We tried to place all EXCEPTION objects in one header file
(exception
// _interface.h) and be used whereever needed, but this resulted in an error
// came from the loader saying they are multiply defined, the same eror came
// out when even we placed them global in the same file, so we had to put each
// in the proper function where the Exception is expected.

//
============================================================================

// An interface to ontos DB operation OC_open(dbname)

void    ada_c_open_database(char* dbname)
{
        ExceptionHandler  db_open_failed ("DatabaseOpenFailed");

        if (db_open_failed.Occurs())
               ada_c_set_exception_code(db_open_failed_code);
        else
  {
   ada_c_set_exception_code( normal_code);

            if (!(OC_dbIsOpen())) OC_open(dbname);
  }
}

//
============================================================================

// An interface to ontos DB operation OC_close(dbname)

void    ada_c_close_database(char* dbname)
{
        ExceptionHandler db_not_open    ("DatabaseNotOpen");

        if (db_not_open.Occurs())
               ada_c_set_exception_code(db_not_open_code);
        else
  {
            ada_c_set_exception_code( normal_code);

            OC_close(dbname);
  }
}

//
============================================================================

// An interface to ontos DB operation OC_transactionStart()

void    ada_c_transaction_start()

// According to Ontos; No Exceptions are associated with this operation
// Howerver the one added below wil catch any exception raised inside ONTOS

{

       ExceptionHandler ontos_falure  ("Failure");
```

```
        if (ontos_falure.Occurs())
                ada_c_set_exception_code(ontos_failure_code);
        else
            {
               ada_c_set_exception_code( normal_code);

            OC_transactionStart();
            }

}

//
==========================================================================

// An interface to ontos DB operation OC_transactionCommit()

void   ada_c_transaction_commit()
{
         ExceptionHandler no_active_transaction ("NoTransaction");

         if (no_active_transaction.Occurs())
                ada_c_set_exception_code(no_active_transaction_code);
         else

            {
             ada_c_set_exception_code( normal_code);

             OC_transactionCommit();
            }
}

//
==========================================================================

// I Think the following 2 operations should be moved to "Component_inteface"
// because both are specific to designer objects and we can not make them
// general to accept any object type

//
==========================================================================

// An interface to ontos DB operation putObject() .. specific to an object

void   ada_c_save_to_db(Component* ada_ptr)

{
  ExceptionHandler  object_already_exists("NameInUse");
  if (object_already_exists.Occurs())
               ada_c_set_exception_code(object_already_exists_code);
  else
      {
        ada_c_set_exception_code( normal_code);

        ada_ptr->putObject();
      }
}

//
```

```
=============================================================================

// An interface to ontos DB operation deleteObject() .. specific to an object


void   ada_c_delete_from_db(Component* ada_ptr)

{
  ExceptionHandler  no_such_object ("NameNotFound");

  if (no_such_object.Occurs())
              ada_c_set_exception_code(no_such_object_code);
  else
     {
       ada_c_set_exception_code( normal_code);

       ada_ptr->deleteObject();
     }
}

//
=============================================================================


/* --------------------------------------------------------------------------

-- Unit             : The header for exception_interface (.h)
-- File   : exception_interface.h
-- Date             : Documented Feb 29, 1996.
-- Author           : Ruey-Wen (Vincent) Hong
-- Systems          : Sun C++ and ONTOS (2.1)
-- Description      : The Ada/C++ Bridge Control Technique is originally created
--                  : by Osman Ibrahim.
--                  :  Provides an interface to a set of exception codes defined
                       below and the protypes for 2 functions that sets and gets
                       the values of an exception code set by differnt functions
                       from db_utility and Component_interface units indicating
                       that some exception has occured or not, the meaning of
                       the exception codes are :

                       normal_code: no exception has occurred
                       object_already_exists_code:  An attempt was made to store
                                   object into the DB but another object is
                                   already exists in DB having the same name
                       no_such_object_code: No such object in the DB
                                   havining that name
                       db_open_failed_code: An attempt was made to open
                                   a DB that does not exist or using a wrong
                                   name (eg misspelled)
                       db_not_open_code: An attempt was made to close
                                   a DB that was not previously opened
                       no_active_transaction_code:   An attempt was made to
                                   commit a transaction that was not started yet

-----------------------------------------------------------------------------
*/

#include <Exception.h>
```

```
#define   normal_code                 0
#define   object_already_exists_code  1
#define   no_such_object_code         2
#define   db_open_failed_code     3
#define db_not_open_code  4
#define no_active_transaction_code 5
#define   ontos_failure_code          6

// Note: The following exception objects have been moved from here to the
//       proper local places; specifically each to a Function(s) inside
//       Component_interface.cxx and db_utility.cxx. The reason for this
//       obligatory movement is because the loader complains of being defined
//       here and used there and gives me " objects so and so are multuply
//       defined". I'm not sure what is wrong because inspite of the error
msg.,
//       the program links and run normally. OSMAN Oct 5, 1995

// ExceptionHandler  db_open_failed ("DatabaseOpenFailed");
// ExceptionHandler  db_open_failed("DatabaseOpenFailed");
// ExceptionHandler  db_not_open("DatabaseNotOpen");
// ExceptionHandler  no_active_transaction ("NoTransaction");
// ExceptionHandler  object_already_exists ("NameInUse");
// ExceptionHandler  no_such_object ("NameNotFound");


// The following function was introduced to allow ada to capture an exception
// raised inside ONTOS so that Ada can handle it in a way taht will not cause
// the program to abort because of a user error; e.g a misspelled DB name.

int ada_c_get_exception_code();

// The following function is used by different operations from inside
// db_utility and Component_interface to set exception code into one of the
// above codes acording to the situation.

void ada_c_set_exception_code(int);

/* ------------------------------------------------------------------------

-- Unit           : The implementation for exception_interface (.cxx)
-- File           : exception_interface.cxx
-- Date           : Documented Feb 29, 1996.
-- Author         : Ruey-Wen (Vincent) Hong
-- Systems        : Sun C++ and ONTOS (2.1)
-- Description    : The Ada/C++ Bridge Control Technique is originally created
--                : by Osman Ibrahim.
--                :  Provides the imlementation for the 2 functions set and
get
                     exception_code that sets and gets the values of an

exception code set by differnt functions from db_utility
                     and Component_interface units indicating that some
exception
                     has occured or not, the meaning of the exception codes
                     are :

                     normal_code: no exception has occurred
                     object_already_exists_code:  An attempt was made to store
                                 object into the DB but another object is
```

```
                    already exists having the same name
          no_such_object_code: No such object in the DB
                    havining that name
          db_open_failed_code: An attempt was made to open
                    a DB that does not exist or using a wrong
                    name (eg misspelled)
          db_not_open_code: An attempt was made to close
                    a DB that was not previously opened
          no_active_transaction_code:   An attempt was made to
                    commit a transaction that was not started yet

----------------------------------------------------------------------------
*/

#include "exception_interface.h"

int global_exception_code = 0 ;

int ada_c_get_exception_code()

// The following function was introduced to allow ada to capture an exception
// raised inside ONTOS so that Ada can handle it in a way taht will not cause
// the program to abort because of a user error; e.g a misspelled DB name.

{
   return global_exception_code;
}

// The following function is used by different operations from inside
// db_utility and Component_interface to set exception code into one of the
// above code acording to the situation.

void ada_c_set_exception_code(int exception_code)

{
global_exception_code = exception_code ;
}
```

APPENDIX C - ONTOS USERS MANUAL

A. QUICK REFERENCE

This manual contains the commands that you will need to use to create and register an ONTOS database in your own directory for use as a CAPS design database.

1) copy the ONTOS schema to the ultimate physical location of your DDB:

```
>cp /usr/local/ONTOS2/ONTOS/ONTOSDB/db/OntosSchema <MyPath>/<MySchema>
```

2) Change the file permissions:

```
>chmod 666 <MyPath>/<MySchema>
```

3) Invoke DBATool (in /usr/local/ONTOS2/bin/DBATool)

```
>DBATool
```

you will get the prompt ">>", enter the following command to register a kernel for your DB

```
>> register kernel <MyKernel> on <Ontos_Server> at <MyPath>/<MySchema>
```

where MyKernel and MySchema are the names of your kernel and physical DB schema respectively. MyPath is simply the path to your schema (the physical location of your DDB). The next 3 commands link the kernel with the schema:

```
 >> register database <MySchema> with kernel <MyKernel>
 >> sync
 >> quit
```

The next command creates a Makefile:

```
  cp $CAPSHOME/doc/DDB_make <MyPath>/Makefile
```

Finally, run the following Makefile command:

```
  >make freshdb
```

NOTE: In some instances, we have observed the DB refreshing process deadlock. If you do not see any progress for more than a minute, kill the process using Control-C and issue the above command again.

After a DDB has been created, it can be erased and reclassified using "make freshdb".

B. MAKEFILE

```
# Logical database name
#
# MODIFY THE NEXT LINE TO THE NAME OF YOUR DB

LOGDBNAME=vincent_db

#
# MODIFY Where your physical database schema file lives

DBFILE=/n/suns5/work/vincent/swbase/MySchema

################################################################
#                                                              #
#          DO NOT MODIFY ANYTHING BELOW THIS POINT             #
#                                                              #
################################################################
#
# Locations of libraries and include files
#
ONTOSINCDIR=  /usr/local/ONTOS2/include/ONTOS
ONTOSLIBDIR= /usr/local/ONTOS2/bin
#
#
# Source files

CAPS_SWB_SRCS = component.cxx db_interface.cxx db_utility.cxx
exception_interface.cxx
#
# Files to classify
#

CLASSIFILES1 = component.h

#
# +X means classify with extensions
EXTENSION = +X
# +v means print what is being classified
VERBOSE = +v
#
# Where OntosSchema lives
ONTOSSCHEMA = /usr/local/ONTOS2/ONTOS/ONTOSDB/db/OntosSchema

LIBS=-L$(ONTOSLIBDIR) -lONTOS
INCLUDES=-I$(ONTOSINCDIR)

CAPS_SWB_OBJS =  $(CAPS_SWB_SRCS:.cxx=.o)


template :
            a.make -v -f *.a
            a.ld -o template.exe test_swb -d -L/usr/local/CC2.1/SC1.0 lONTOS -
l /usr/local/CC2.1/SC1.0/patch  template.exe
```

```
.SUFFIXES: .cxx

.cxx.o:
            CC -c -go $(INCLUDES) *.cxx


classify1:
            classify +D$(LOGDBNAME) $(EXTENSION) $(VERBOSE) \
            $(INCLUDES) $(CLASSIFILES1)


freshdb:
            rm $(DBFILE)*
            cp $(ONTOSSCHEMA) $(DBFILE)
            chmod 777 $(DBFILE)
            make classify1


caps:
            cplus -c -g -Bstatic $(CAPS_SWB_SRCS) $(INCLUDES) $(LIBS)

component:
            cplus -c -g -Bstatic component.h $(INCLUDES) $(LIBS)
            CC -c -go $(INCLUDES) component.cxx
```

APPENDIX D - SOFTWARE BASE USERS MANUAL

To use the CAPS software base, the user has to start CAPS by typing "caps" at the prompt. If your CAPS environment has been set up correctly, you will get the CAPS main menu shown in Figure 14. If the CAPS main menu does not come up correctly, you should inform your CAPS manager to check on your CAPS environment configuration.

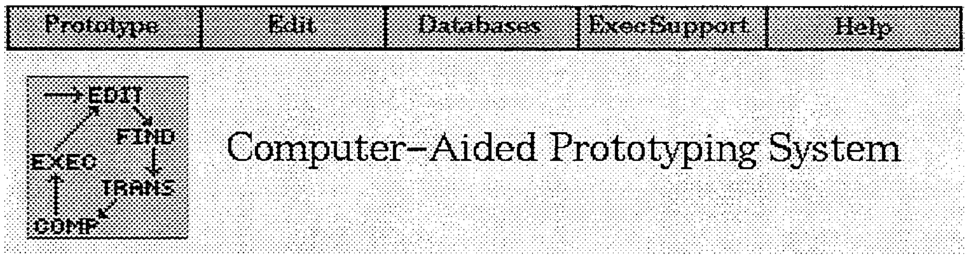


Figure 14. CAPS Start Up Window

To invoke the CAPS Software Base, click on the Database selection square with the left mouse button. Two submenus will be displayed: Design Database and Software Base. Here we want to explore the Software Base, so left click the mouse on the Software Base selection.

Figure 15 will come up within a few seconds. From this point on, you will interact directly with the Software Base utility.

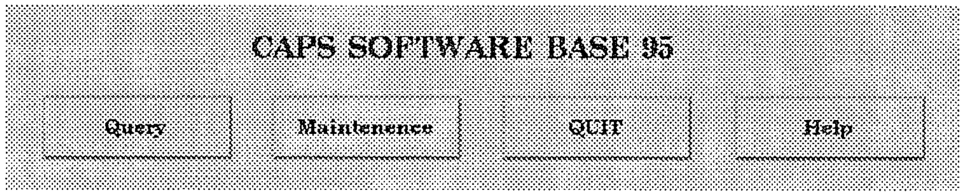


Figure 15. CAPS Software Base Main Menu

As we mentioned earlier, the SBGUI can be separated into two parts(User part and DBMA part), with the submenu structures shown in Figure 16.

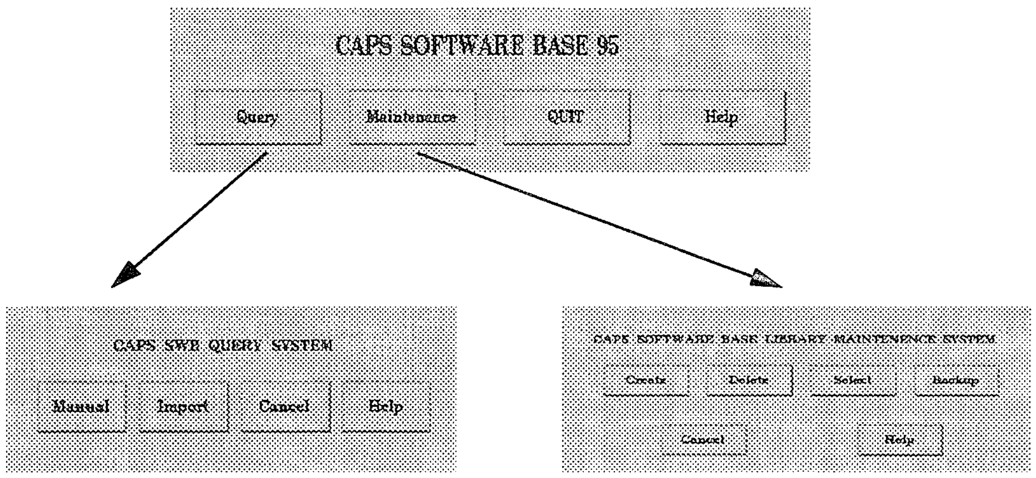


Figure 16. Structure of the User and DBMA Options

Access to the Query option is available to all users. Whether a user can select the Maintenance option depends on the CAPS user mode. The Software Base will check the user's privilege to see whether s/he is a CAPS manager. When the user is a CAPS manager, s/he has access to the Maintenance option, otherwise the user can only execute the query option. An unauthorized click on the maintenance button will be rejected. As a user manual, this document concentrates on the operational assistance within the Software Base.

A. THE QUERY OPTION

Let us start from the query option. When the user left click the Query button, the query window shown in Figure 17 is displayed.

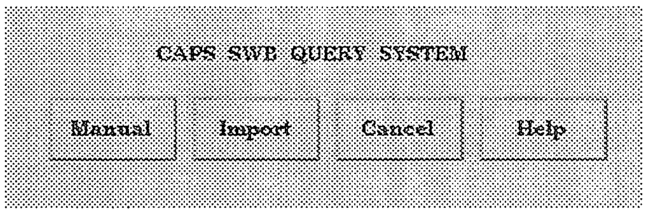
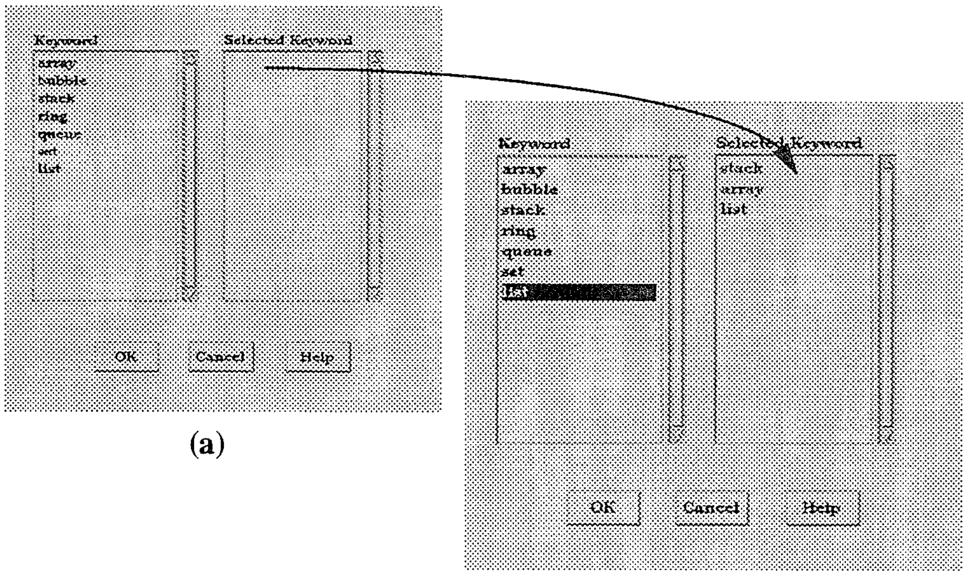


Figure 17. CAPS Software Base Query Menu

Query functionality can be separated into Manual and Import. Manual means that the user has to input the query specification manually. Import identifies the capability of searching the software library based on the specification automatically generated from the SDE.

When the user left click the Manual button, it will trigger the window shown in Figure 18(a). The user should select the keywords s/he needs from the “Keyword” window and the system will display them on the right hand side, “Selected Keyword” window automatically. After the selection, Figure 18(b) will be displayed.

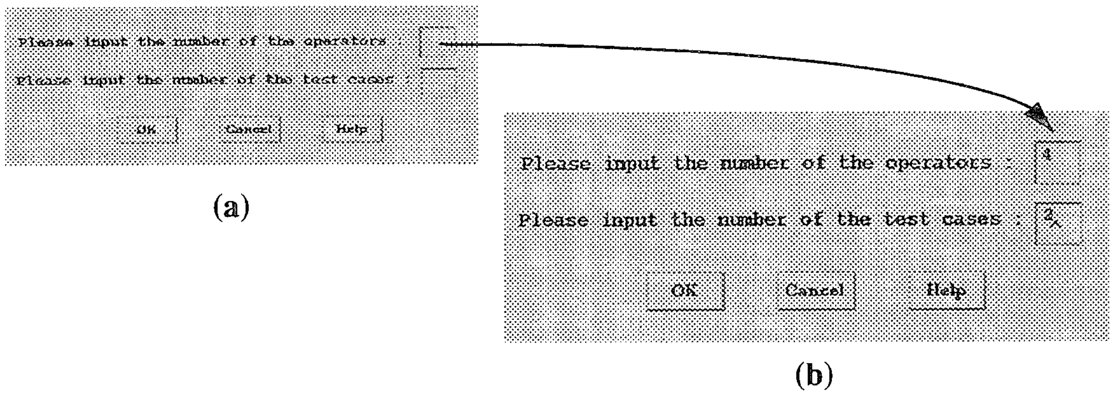


(a)

(b)

Figure 18. Keyword Selection Window

When the user finishes with Keyword selection, s/he left clicks the OK button. The system will keep these selected keywords in a predefined Query_Keywords_String and then trigger the window in Figure 19(a) below. It allows the user to input the number of query operators and the number of test cases which are the two critical requirements to the multi-level filtering retrieval technique(MLFRT).



Please input the number of the operators :

Please input the number of the test cases :

(a)

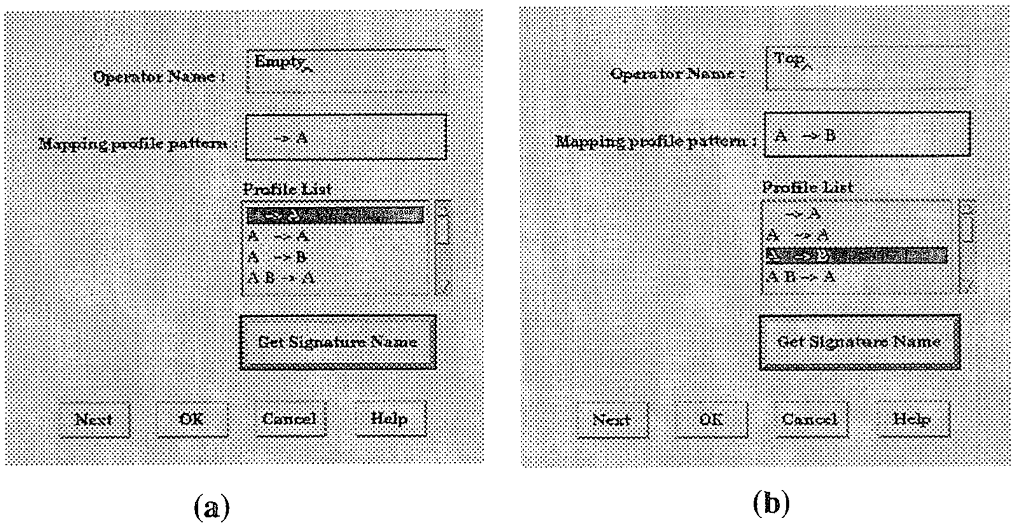
Please input the number of the operators : 4

Please input the number of the test cases : 2

(b)

Figure 19. Get the Number of Operators and Test Cases

After entering the number of operators and test cases(shown in Figure 19(b) displayed), the system will display the window in Figure 20.



Operator Name : Empty

Mapping profile pattern : -> A

Profile List

-> A
A -> A
A -> B
A B -> A

(a)

Operator Name : Top

Mapping profile pattern : A -> B

Profile List

-> A
A -> A
A -> B
A B -> A

(b)

Figure 20. Get Detail Information for Each Operator

In order that the desired component be matched as closely as possible, all of its internal operators must be fully specified. Full specification is tied to these internal operator's names and profile patterns.

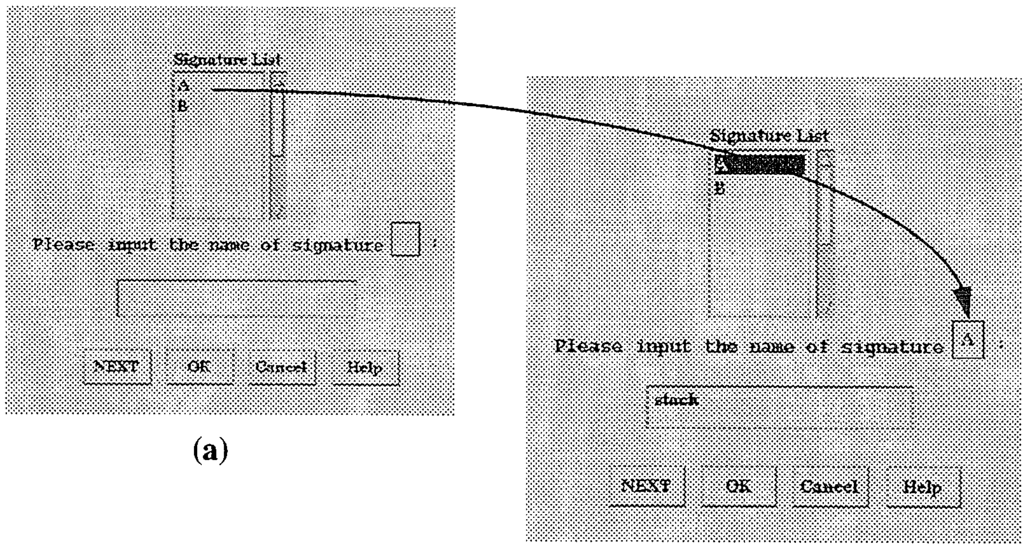
The name is simply a description of what the user believes the internal operator does. It is used in part of the later matching process.

The profile description is a little more complicated and is broken into two stages, giving rise to the name multi-level filtering. The first step is to define the inputs and outputs in a general level. This is the operator's profile. For example, if looking for a component such as a stack, user need to specify an Empty operator, a Top operator, a Push operator, etc.

Figure 20 shows how these internal operators are specified. This must be done for each internal operator of the desired component. Figure 20 will be repeatedly popped up for the number of the operators which user input in Figure 19. Figure 20(a) and Figure 20(b) are two examples showing input for the operator Empty and Top when the user searches for a Stack-like software component. In the MLFRT process, the user needs to supply the query information for *operator name*, *profile* and the *signature name.*

Notice the profile selected in Figure 20(b). It shows that there is one input and one output(A -> B). If for example, there were two inputs and one output of different types, user should pick "A B -> C".

Once this profile pattern is specified, user must get the *signature names*, which are fine grained specifications for the unique input and output signatures. Continuing with the Top internal operator, Figure 21 shows how this information is gathered. Once the user clicks on a signature, it is displayed in the box as shown in Figure 21(b).



Please input the name of signature A :

stack

(a)

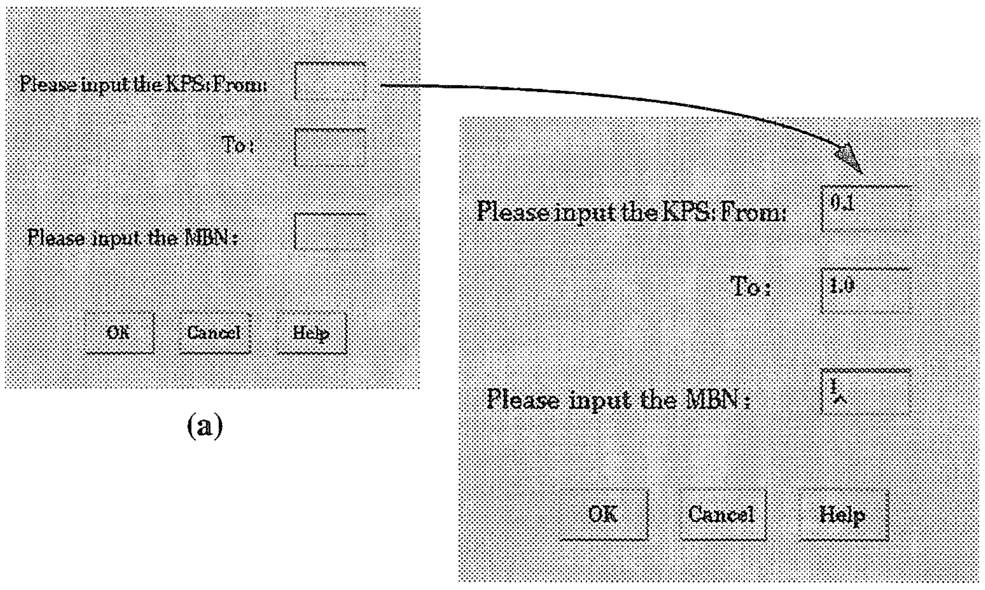
(b)

Figure 21. Get the Name for each Signature in the Profile

The user then enters a signature name which matches its type as close as possible. Once this is done, the Next button is left clicked to commit to name.

Obviously, if the user tries to find a stack component, s/he must specify the other internal operators like the Top operator in the example above. The Next button in Figure 20 is selected after completing the specifications of each of these operators. Once all internal operators of the desired component are specified, the OK button in Figure 20 is selected. The next step is to decide how close the match must be for a component to be selected as a possible choice.

Figure 22 shows how the matching criteria is constrained. The KPS(Keyword, Profile, Signature Match Ratio) entries defines a lower and upper bound on how closely the candidate components matches the desired component. Also shown in Figure 22 is an entry for MBN(Map Block Number). It is used to constrain the display of the candidate components. Figure 22(b) shows an example which user specify the KPS is from 0.1 to 1.0 and the MBN is 1.


(a)

Please input the KPS:From:

To:

Please input the MBN:

(b)

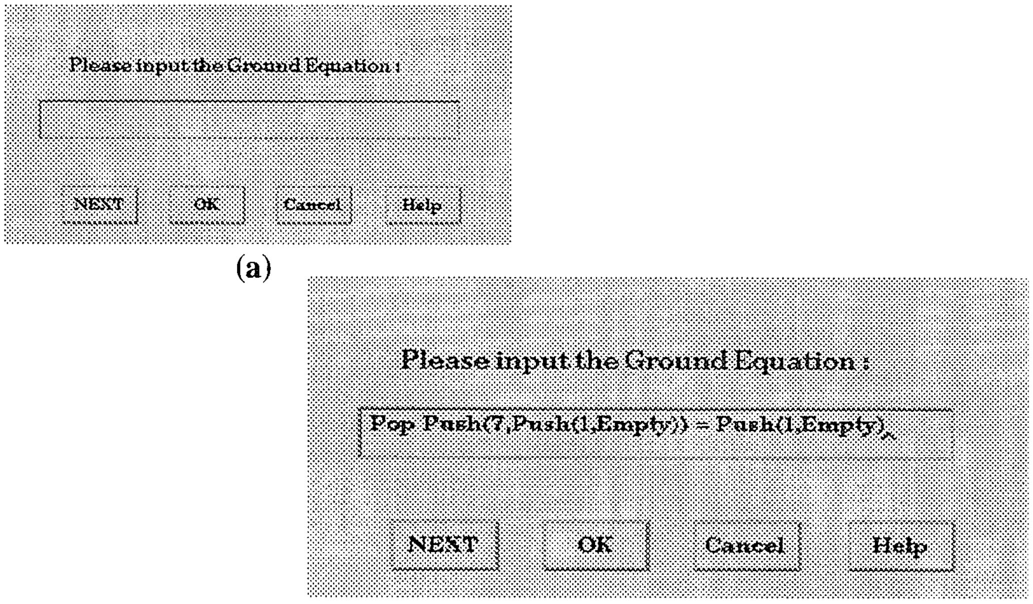
Please input the KPS:From: 0.1

To: 1.0

Please input the MBN: 1

Figure 22. Get the KPS and MBN

The final step prior to doing the search is specifying the Ground Equation which is required by the MLFRT(shown in Figure 23(a).


Please input the Ground Equation :

Please input the Ground Equation :

Pop Push(7,Push(1,Empty)) = Push(1,Empty)

(b)

Figure 23. Get Ground Equation for OBJ3

The Ground Equations are needed for the semantic behavior matching of all the internal operators previously specified. These equations are built for utilizing the OBJ3 Algebra Specification Language. If the user is not familiar with this language, outside help will be required as that is beyond the scope of this manual. An example of a Ground Equation for a Stack-like component query is displayed in figure 23(b).

The Next button of Figure 23 has similar functionality as the Next button did in Figure 21. It will clear the ground equation keyin buffer and let the user keyin the next ground equation. After all the information is entered, left click the OK button and the window displayed in Figure 24 will appear.

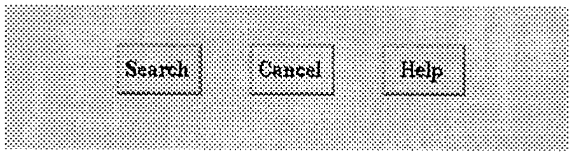


Figure 24. Start Search Panel

The final step in the query process is to push the Search button and the search will be executed. The system will feed the query specification into some particular query form, and feed them into the Multi-Level Filtering Retrieval Module which uses a DFS algorithm to look up the software base and retrieves the proper candidate components.

Here is a practical example:

Suppose that a user submits a query containing the following information:

- The keywords are: Booch, Data-Structure, Stack.
- The partial specification is:

```
Package Stack-Of-Nat is Type Stack;
   -- Query operations --
   function Empty(Out: Stack) .
   function Top(In: Stack; Out: Nat) .
   function Push(In: Nat, Stack; Out: Stack) .
   function Pop(In: Stack; Out: Stack) .
   -- Query ground equations --
   Top Push(1,Empty) = Top Pop(Push(6,Push(1,Empty))) .
   Pop Push(7,Push(1,Empty)) = Push(1,Emty) .
End of Package Stack-Of-Nat;
```

- The selection criterion is to choose component that has a KPS from 0.1 to 1.0. The MBN is selected to be 1.

Given this query, the program scan the operations above to compute the profile table

The result which comes out will include following information:

```
--------------------------------------------------------------------------
Find component: Stack2.obj
Using Map Number:        1
KeywordRank:     1.0E+00, ProfileRank:    1.0E+00
SignatureRank:    1.0E+00, SemanticRank:   4.0E+00
The ComponentRank Value:    4.0E+00
--------------------------------------------------------------------------
Find component: list.obj
Using Map Number:        1
KeywordRank:     6.7E-01, ProfileRank:    1.0E+00
SignatureRank:    1.0E+00, SemanticRank:   4.0E+00
```

```
The ComponentRank Value:   2.7E+00
------------------------------------------------------------------------
Find component: queue.obj
Using Map Number:          2
KeywordRank:      6.7E-01, ProfileRank:    1.0E+00
SignatureRank:    1.0E+00, SemanticRank:   2.0E+00
The ComponentRank Value:   1.3E+00
------------------------------------------------------------------------
Find component: bag.obj
Using Map Number:          3
KeywordRank:      6.7E-01, ProfileRank:    1.0E+00
SignatureRank:    1.0E+00, SemanticRank:   0.0E+00
The ComponentRank Value:   0.0E+00
------------------------------------------------------------------------
```

The query produced four candidate components which are all very similar. The output is listed in order of matching degree. Notice that Stack2.obj has a higher ComponentRank Value than all the rest and therefore is probably the best candidate for the query. It is now up to the user to decide which one s/he wants, or go back and redo the query with tighter specification requirements.

B. THE MAINTENANCE OPTION

When maintaining the software library, the CAPS manager has to push the Maintenance button shown previously in Figure 14. Software Base system will pop up as shown in Figure 25.

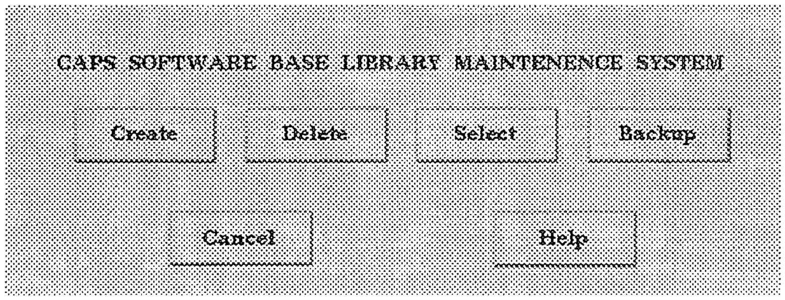


Figure 25. CAPS Software Base Maintenance Menu

According to the functional requirements of the SBGUI, software base maintenance work includes library creation, selection, deletion, and backup. After the DBMA selects a desired library, s/he should be able to append, delete, modify, and browse software components.

APPENDIX E - SBGUI TAE SOURCE CODE

```
-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.
-- *** File:         global_s.a
-- *** Generated:    Feb 20 10:57:53 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ***************************************************************************
--  *
--  *   Global                                -- Package SPEC
--  *
--  ***************************************************************************

with X_Windows;
with Text_IO;
with TAE;

with SYSTEM;
use TAE,SYSTEM,Text_IO;

package Global is

--| PURPOSE:
--| This package is automatically "with"ed in to each panel package body.
--| You can insert global variables here.
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| This file is generated only once.  Therefore, you may modify it without
--| impacting automatic code merge.
--|
--| CHANGE LOG:
--| 20-Feb-96   TAE     Generated

  package Taefloat_IO is new Text_IO.Float_IO (TAE.Taefloat);

  Default_Display_Id : X_Windows.Display;

--ADDED

-----------------------------------------------------
-- ADDED By Ruey-Wen (Vincent) Hong
-- This part is only for Pan_Opdetail_b.a
-----------------------------------------------------

  type sig_chk_type is array(1..10) of string(1..1);
  Sig_Exist_Chk_Vec   : sig_chk_type :=
("0","0","0","0","0","0","0","0","0","0");
```

```
--  type sig_keep_vec_type is array(1..10) of string(1..1);
--  Sig_Keep_Vec : sig_keep_vec_type := (others => " ");

  Sig_Keep_Vec : array(1..10) of string(1..1) := (others => " ");

--    Sig_Keep_Vec : array (1..1) of String (1..10);

--    Sig_Keep_Vec : String(1..10) := (others => " ");

  skv_index : integer := 1;

  kps_from : float;
  kps_to   : float;
  mbn      : natural;


-- MODIFIED/ADDED

  library             : String (1..10):= ('A','d','a',others=>' ');
  COMPONENT_IS_OPERATOR : BOOLEAN;

  lib_to_delete       : String (1..10):= (others=>' ');
  path                : String(1..80):= (others=>' ');
  proto_prefix        : String(1..80):= (others=>' ');
  Query_psdl          : String(1..80):= (others=>' ');
  Directory           : String(1..80):= (others=>' ');
  kwquery_outfile     : String(1..15):="kwquery_outfile";
  query_outfile       : String(1..13):="query_outfile";
  component           : String(1..80):= (others=>' ');
  directory_array     : String (1..27):= (others=>' ');
  directory_file_name: String(1..14)  := "directory_file";
  lib_vec             : s_vector(1..20):= (others=> new STRING(1..10));
  file_vec            : s_vector(1..1000):= (others=> new STRING(1..80));
  Is_a_directory      : Boolean:=FALSE;
  Upper_directory     : Boolean:=FALSE;
  Component_add       : Boolean:=FALSE;
  Component_update    : Boolean:=FALSE;
  Query               : Boolean:=FALSE;
  current_directory   : array(1..20) of String(1..80);
  directory_file      : file_type;
  cur_dir_index       : integer:=1;
  lib_count           : integer:=0;
  num_of_comp         : integer:=1;

-- MODIFIED
  com                 : constant String:=
                          "/n/suns7/work/caps93/SUN4/bin/sb ";

  parse               : constant String:="/n/sun54/work/caps92/src/
software_base/integrate/";
  gui_directory       : constant String:="/n/sun54/work/caps92/src/
user_interface/sb_interface/";
```

```
--ADDED
  OPERATOR_FILE_NAME : constant STRING := "operator_psdl_spec.txt";
  TYPE_FILE_NAME     : constant STRING := "type_psdl_spec.txt";
  PROTOTYPE_FILE_NAME : constant STRING := "prototype_psdl_spec.txt";
  com2               : constant String:=
                         "/n/suns7/work/caps93/SUN4/bin/opsig ";
  com3               : constant String:=
                         "/n/suns7/work/caps93/SUN4/bin/adtsig ";
--  THE_STATUS              : COMPONENT_STATUS;
  HAS_GENERICS            : BOOLEAN;
  QC_IN_PARAMS,
  QC_OUT_PARAMS,
  SBC_IN_PARAMS,
  SBC_OUT_PARAMS,
--  GEN_PARAMS              : PARAMETERS;
  QC_IN_COUNT,
  QC_OUT_COUNT,
  SBC_IN_COUNT,
  SBC_OUT_COUNT,
  GEN_COUNT               : INTEGER;
--  SBC_NAME,
--  QC_NAME                 : PSDL_ID_PKG.PSDL_ID;




  --  ...................................................................
  --  .
  --  .    Application_Done                 -- Subprogram SPEC
  --  .
  --  ...................................................................

  function Application_Done
    return Boolean;

  --| PURPOSE:
  --| This function returns true if a "quit" event handler has called
  --| Set_Application_Done, otherwise it returns false.
  --|
  --| EXCEPTIONS: (none)
  --|
  --| NOTES: (none)




  --  ...................................................................
  --  .
  --  .    Set_Application_Done             -- Subprogram SPEC
  --  .
  --  ..........................................................................

  procedure Set_Application_Done;

  --| PURPOSE:
  --| This procedure can be used by an event handler, typically a "quit"
  --| button, to signal the end of the application.
  --|
  --| EXCEPTIONS: (none)
  --|
  --| NOTES: (none)
 --ADDED

  --............................................................................
  --  .
  --  .    system_call                         -- Subprogram SPEC
  --  .
  --  ..........................................................................

  procedure system_call(command : STRING);

  --| PURPOSE:
  --| This procedure is used to make unix system calls from within the
program.

  --............................................................................
  --  .
  --  .    strlen                              -- Subprogram SPEC
  --  .
  --  ..........................................................................

  procedure strlen(s: in String; n: in out Integer);

  --| PURPOSE:
  --| This procedure is used to get the length of strings.

  --............................................................................
  --  .
  --  .    list_directory                      -- Subprogram SPEC
  --  .
  --  ..........................................................................

  procedure list_directory(file     :in out file_type;

file_name:in out string;

file_vec :in out s_vector;
                                                                              I
:in out integer);

  --| PURPOSE:
  --| This procedure is used to obtain the contents of unix directory
structures.
```

```
  --..............................................................
  --  .
  --  .    list_components                 -- Subprogram SPEC
  --  .
  --  ............................................................

  procedure list_components(file     :in out file_type;

file_name:in out string;

file_vec :in out s_vector;
                                                                I
:in out integer);

  --| PURPOSE:
  --| This procedure is used to read the component list from a text file and
  --| fill them into a s_vector structure to be displayed in a TAE panel.

  --..............................................................
  --  .
  --  .    read_directory                  -- Subprogram SPEC
  --  .
  --  ............................................................

  procedure read_directory(file     :in out file_type;

file_name:in out string;

dir_name :in out string);

  --| PURPOSE:
  --| This procedure is used to read the name of the current directory and
  --| to get the path from a text file.

  --..............................................................
  --  .
  --  .    errorstring                     -- Subprogram SPEC
  --  .
  --  ............................................................

  procedure errorstring(file     :in out file_type;

file_name:in out string;

err_str  :in out string);

  --| PURPOSE:
  --| This procedure is used to read the error message given by the software
  --| base program.

  --..............................................................
  --  .
  --  .    parse_line                      -- Subprogram SPEC
  --  .
  -- .............................................................

  procedure parse_line(s: in String);

  --| PURPOSE:
  --| This procedure is used to determine if the selected line is a directory
  --| or a file and if it is a directory it gets the identity of the
directory.


end Global;


-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        global_b.a
-- *** Generated:   Feb 20 10:57:53 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  **************************************************************************
--  *
--  *   Global                                -- Package BODY
--  *
--  **************************************************************************

package body Global is

--| NOTES: (none)
--|
--| REGENERATED:
--| This file is generated only once.  Therefore, you may modify it without
--| impacting automatic code merge.
--|
--| CHANGE LOG:
--| 20-Feb-96   TAE     Generated

  Is_Application_Done : Boolean := FALSE;

  --  .............................................................
  --  .
  --  .   Application_Done                    -- Subprogram BODY
  --  .
  --  .............................................................

  function Application_Done
    return Boolean is

  --| NOTES: (none)

  begin -- Application_Done

    return Is_Application_Done;
```

```
end Application_Done;


--  ..................................................................
--  .
--  .    Set_Application_Done              -- Subprogram BODY
--  .
--  ..................................................................

procedure Set_Application_Done is

--| NOTES: (none)

begin -- Set_Application_Done

  Is_Application_Done := TRUE;

end Set_Application_Done;


-----------------------------------------------------------------------
--ADDED By RUEY-WEN HONG  960110
-----------------------------------------------------------------------
--  ..................................................................
--  .
--  .     system_call                       -- Subprogram BODY
--  .
--  ..................................................................

procedure system_call(command : STRING) is

          procedure system_c (command : ADDRESS);
          pragma INTERFACE(C, SYSTEM_C);
          pragma INTERFACE_NAME(SYSTEM_C, "_system");
          TEMP : constant STRING := command&ASCII.NUL;
          ERROR : INTEGER;

begin
          SYSTEM_C(TEMP'ADDRESS);
end system_call;

--  ..................................................................
--  .
--  .     strlen                            -- Subprogram BODY
--  .
--  ..................................................................

procedure strlen(s: in String; n: in out Integer) is

          I : Integer := 1;

begin
          loop
           if s(I) = ' ' then
```

```
              exit;
             end if;
             I:=I+1;
    end loop;
            n:= I-1;
  end strlen;

  --  ...................................................................
  --  .
  --  .     list_directory                 -- Subprogram BODY
  --  .
  --  ...................................................................

  procedure list_directory(file:in out file_type;

file_name:in out string;

file_vec : in out s_vector;
                          I :in out integer)is
          len : integer :=1;

  begin

          I:=1;
    open(file,mode=>in_file,name=>file_name);
    file_vec(I).all(1..2):="..";
          for cl in 3..80 loop
            file_vec(I).all(cl):=' ';
    end loop;
          while not end_of_file (file) loop
            I:=I+1;
            Text_IO.get_line(file,file_vec(I).all,len);
            for clean in (len+1)..80 loop
                    file_vec(I).all(clean):=' ';
      end loop;
    end loop;
    close(file);
          exception
            when END_ERROR => null;

  end list_directory;

  --  ...................................................................
  --  .
  --  .    list_components
  --  .
  --  ...................................................................

  procedure list_components(file     :in out file_type;

file_name:in out string;

file_vec : in out s_vector;
                                                                      I
:in out integer)is
    len: integer :=1;

  begin

            I:=1;
            open(file,mode=>in_file,name=>file_name);
            while not end_of_file (file) loop
              Text_IO.get_line(file,file_vec(I).all,len);
              for clean in (len+1)..80 loop
                        file_vec(I).all(clean):=' ';
      end loop;
              I:=I+1;
    end loop;
            I:=I-1;
            close(file);
  end list_components;

  --  ..................................................................
  --  .
  --  .     read_directory
  --  .
  --  ..................................................................

  procedure read_directory(file     :in out file_type;

file_name:in out string;

dir_name :in out string) is
    len: integer :=1;

  begin
            open(file,mode=>in_file,name=>file_name);
    Text_IO.get_line(file,dir_name,len);
            close(file);
  end read_directory;

   --  ..................................................................
   --  .
   --  .     errorstring
   --  .
   --  ..................................................................

  procedure errorstring(file     :in out file_type;
                        file_name:in out string;

err_str  :in out string)is
    len: integer :=1;

  begin
            open(file,mode=>in_file,name=>file_name);
            Text_IO.get_line(file,err_str,len);
    if len=0 then
              Text_IO.get_line(file,err_str,len);
    end if;
            close(file);
            exception
              when END_ERROR =>  close(file);
  end errorstring;

  --  ...................................................................
  --  .
  --  .     parse_line
  --  .
  --  ...................................................................

  procedure parse_line(s: in string) is

            char:character;
    N   :integer:=1;

  begin

            strlen(s,N);
    if s(N)='/' then
              Is_a_directory:=TRUE;
    end if;
    if N=2 then
              if s(1..2)=".." then
                Upper_directory:=TRUE;
      end if;
    end if;
  end parse_line;


end Global;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_backup_1_s.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author     : Ruey-Wen (Vincent) Hong
-- *** Date       : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ************************************************************************
--  *
--  *    Panel_backup_1                    -- Package SPEC
--  *
--  ************************************************************************

with TAE;
with X_Windows;

package Panel_backup_1 is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  backup_1
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  backup_1
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;    -- panel information


  --  ......................................................................
  --  .
  --  .    Initialize_Panel                     -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Initialize_Panel
    ( Collection_Read                        -- TAE Collection read from
        : in TAE.Tae_Co.Collection_Ptr );    -- resource file

  --| PURPOSE:
  --| This procedure initializes the Info.Target and Info.View for this panel
  --|
  --| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
  --| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
  --|    Collection_Read
  --|
  --| NOTES: (none)




  --  ......................................................................
  --  .
  --  .    Create_Panel                         -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Create_Panel
    ( Panel_State                            -- Flags sent to Wpt_NewPanel.
        : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                         -- Panel origin is offset from
      : in X_Windows.Window                 -- this X Window.  Null_Window
        := X_Windows.Null_Window );         -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--   ...................................................................
--   .
--   .    Connect_Panel                     -- Subprogram SPEC
--   .
--   ...................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                         -- Panel origin is offset from
      : in X_Windows.Window                 -- this X Window.  Null_Window
        := X_Windows.Null_Window );         -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--   ...................................................................
--   .
```

```
  --  .    Destroy_Panel                    -- Subprogram SPEC
  --  .
  --  ..............................................................

  procedure Destroy_Panel;

  --| PURPOSE:
  --| This procedure erases a panel from the screen and de-allocates the
  --| associated panel object (not the target and view).
  --|
  --| EXCEPTIONS:
  --| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
  --|
  --| NOTES:
  --| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
  --| in any Wpt call until it is created again.




  --  ..............................................................
  --  .
  --  .    Dispatch_Item                    -- Subprogram SPEC
  --  .
  --  ..............................................................

  procedure Dispatch_Item
    ( User_Context_Ptr                     -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_backup_1;


-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         pan_backup_1_b.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : By Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  **************************************************************************
--  *
--  *    Panel_backup_1                    -- Package BODY
--  *
--  **************************************************************************
```

```
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_backup_1 is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  backup_1
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   backup_message,  full_backup,    partial_backup
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

  --  .......................................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram BODY
--  .
--  ...................................................................

procedure Initialize_Panel
  ( Collection_Read
      : in TAE.Tae_Co.Collection_Ptr ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

begin -- Initialize_Panel

  Info := new TAE.Tae_Wpt.Event_Context;
  Info.Collection := Collection_Read;
  TAE.Tae_Co.Co_Find (Info.Collection, "backup_1_v", Info.View);
  TAE.Tae_Co.Co_Find (Info.Collection, "backup_1_t", Info.Target);

  -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_backup_1.Initialize_Panel:  "
      & "Collection_Read not initialized.");
    raise;

  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_backup_1.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  ...................................................................
--  .
--  .    Create_Panel                         -- Subprogram BODY
--  .
--  ...................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)
```

```
  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id            => Global.Default_Display_Id,
        Data_Vm               => Info.Target,
        View_Vm               => Info.View,
        Relative_Window       => Relative_Window,
        User_Context          => Info,
        Flags                 => Panel_State,
        Panel_Id              => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (backup_1) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_backup_1.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_backup_1.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  ......................................................................
--  .
--  .    Connect_Panel                      -- Subprogram BODY
--  .
--  ......................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)
```

```
  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window       => Relative_Window,
        Panel_State           => Panel_State );
  else
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_STATE =>
    Text_IO.Put_Line ("Panel_backup_1.Connect_Panel:  "
      & "Invalid panel state.");
    raise;

end Connect_Panel;



--  ..........................................................
--  .
--  .    Destroy_Panel                     -- Subprogram BODY
--  .
--  ..........................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_backup_1.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;
  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;



--++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--   .....................................................................
--   .
--   .    backup_message_Event            -- Subprogram SPEC & BODY
--   .
--   .....................................................................

procedure backup_message_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: backup_message.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: backup_message.

begin -- backup_message_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel backup_1, parm backup_message: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: backup_message.
  -- MERGE NOTE: Add code ABOVE this line for parm: backup_message.

end backup_message_Event;



--   .....................................................................
--   .
--  .    full_backup_Event                  -- Subprogram SPEC & BODY
--  .
--  ......................................................................

procedure full_backup_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: full_backup.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: full_backup.

begin -- full_backup_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel backup_1, parm full_backup: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: full_backup.
  -- MERGE NOTE: Add code ABOVE this line for parm: full_backup.

end full_backup_Event;



--  ......................................................................
--  .
--  .    partial_backup_Event               -- Subprogram SPEC & BODY
--  .
--  ......................................................................

procedure partial_backup_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: partial_backup.
```

```
    -- MERGE NOTE: Add declarations ABOVE this line for parm: partial_backup.

  begin -- partial_backup_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel backup_1, parm partial_backup: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: partial_backup.
    -- MERGE NOTE: Add code ABOVE this line for parm: partial_backup.

  end partial_backup_Event;



  --
  -- end EVENT HANDLERS
  --
  ----------------------------------------------------------------------------



  --  ...................................................................
  --  .
  --  .    Dispatch_Item                       -- Subprogram BODY
  --  .
  --  ...................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| NOTES: (none)

  begin -- Dispatch_Item

    if TAE.Tae_Misc.s_equal ("backup_message", User_Context_Ptr.Parm_Name)
then
      backup_message_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("full_backup", User_Context_Ptr.Parm_Name)
then
      full_backup_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("partial_backup",
User_Context_Ptr.Parm_Name) then
      partial_backup_Event (User_Context_Ptr);
    end if;

  end Dispatch_Item;
```

```
-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_backup_1;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:          pan_create_1_s.a
-- *** Generated:     Mar  4 14:14:09 1996
-- *** Author       : Ruey-Wen (Vincent) Hong
-- *** Date         : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ***************************************************************************
--  *
--  *     Panel_create_1                          -- Package SPEC
--  *
--  ***************************************************************************

with TAE;
with X_Windows;

package Panel_create_1 is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  create_1
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  create_1
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;   -- panel information


  --  ......................................................................
  --  .
  --  .    Initialize_Panel                     -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Initialize_Panel
  ( Collection_Read                          -- TAE Collection read from
      : in TAE.Tae_Co.Collection_Ptr );      -- resource file

--| PURPOSE:
--| This procedure initializes the Info.Target and Info.View for this panel
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
--| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
--|    Collection_Read
--|
--| NOTES: (none)




--   ......................................................................
--   .
--   .    Create_Panel                       -- Subprogram SPEC
--   .
--   ......................................................................

procedure Create_Panel
  ( Panel_State                              -- Flags sent to Wpt_NewPanel.
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                          -- Panel origin is offset from
      : in X_Windows.Window                  -- this X Window.  Null_Window
        := X_Windows.Null_Window );          -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--   ......................................................................
--   .
--   .    Connect_Panel                      -- Subprogram SPEC
--   .
--   ......................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;
    Relative_Window                          -- Panel origin is offset from
      : in X_Windows.Window                  -- this X Window.  Null_Window
        := X_Windows.Null_Window );          -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--  ......................................................................
--  .
--  .    Destroy_Panel                       -- Subprogram SPEC
--  .
--  ......................................................................

procedure Destroy_Panel;

--| PURPOSE:
--| This procedure erases a panel from the screen and de-allocates the
--| associated panel object (not the target and view).
--|
--| EXCEPTIONS:
--| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
--|
--| NOTES:
--| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
--| in any Wpt call until it is created again.




--  ......................................................................
--  .
--  .    Dispatch_Item                       -- Subprogram SPEC
--  .
--  ......................................................................

procedure Dispatch_Item
```

```
    ( User_Context_Ptr                        -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_create_1;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_create_1_b.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author     : Ruey-Wen (Vincent) Hong
-- *** Date       : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ************************************************************************
--  *
--  *   Panel_create_1                      -- Package BODY
--  *
--  ************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_create_1 is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  create_1
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   create_1_disp,   create_1_keyin,  create_1_label
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

  --  ...............................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram BODY
  --  .
  --  ...............................................................

  procedure Initialize_Panel
    ( Collection_Read
        : in TAE.Tae_Co.Collection_Ptr ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

  begin -- Initialize_Panel

    Info := new TAE.Tae_Wpt.Event_Context;
    Info.Collection := Collection_Read;
    TAE.Tae_Co.Co_Find (Info.Collection, "create_1_v", Info.View);
    TAE.Tae_Co.Co_Find (Info.Collection, "create_1_t", Info.Target);

    -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

  exception

    when TAE.UNINITIALIZED_PTR =>
      Text_IO.Put_Line ("Panel_create_1.Initialize_Panel:  "
        & "Collection_Read not initialized.");
      raise;
  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_create_1.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  ......................................................................
--  .
--  .    Create_Panel                      -- Subprogram BODY
--  .
--  ......................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id            => Global.Default_Display_Id,
        Data_Vm               => Info.Target,
        View_Vm               => Info.View,
        Relative_Window       => Relative_Window,
        User_Context          => Info,
        Flags                 => Panel_State,
        Panel_Id              => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (create_1) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_create_1.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;
  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_create_1.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  ......................................................................
--  .
--  .    Connect_Panel                    -- Subprogram BODY
--  .
--  ......................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window       => Relative_Window,
        Panel_State           => Panel_State );
  else
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_STATE =>
    Text_IO.Put_Line ("Panel_create_1.Connect_Panel:  "
      & "Invalid panel state.");
    raise;

end Connect_Panel;



--  ......................................................................
--   .
--   .    Destroy_Panel                      -- Subprogram BODY
--   .
--   ...................................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_create_1.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;


--++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--   ...................................................................
--   .
--   .    create_1_disp_Event                -- Subprogram SPEC & BODY
--   .
--   ...................................................................

procedure create_1_disp_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: create_1_disp.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: create_1_disp.

begin -- create_1_disp_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel create_1, parm create_1_disp: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: create_1_disp.
  -- MERGE NOTE: Add code ABOVE this line for parm: create_1_disp.

end create_1_disp_Event;



--  .......................................................................
--  .
--  .    create_1_keyin_Event              -- Subprogram SPEC & BODY
--  .
--  .......................................................................

procedure create_1_keyin_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: create_1_keyin.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: create_1_keyin.

begin -- create_1_keyin_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel create_1, parm create_1_keyin: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;
  -- MERGE NOTE: Add code BELOW this line for parm: create_1_keyin.
  -- MERGE NOTE: Add code ABOVE this line for parm: create_1_keyin.

end create_1_keyin_Event;



--  .....................................................................
--  .
--  .    create_1_label_Event             -- Subprogram SPEC & BODY
--  .
--  .....................................................................

procedure create_1_label_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: create_1_label.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: create_1_label.

begin -- create_1_label_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel create_1, parm create_1_label: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: create_1_label.
  -- MERGE NOTE: Add code ABOVE this line for parm: create_1_label.

end create_1_label_Event;



--
-- end EVENT HANDLERS
--
-------------------------------------------------------------------------



--  .....................................................................
--  .
```

```
  --  .    Dispatch_Item                        -- Subprogram BODY
  --  .
  --  ..........................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| NOTES: (none)

  begin -- Dispatch_Item

    if TAE.Tae_Misc.s_equal ("create_1_disp", User_Context_Ptr.Parm_Name) then
      create_1_disp_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("create_1_keyin",
User_Context_Ptr.Parm_Name) then
      create_1_keyin_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("create_1_label",
User_Context_Ptr.Parm_Name) then
      create_1_label_Event (User_Context_Ptr);
    end if;

  end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_create_1;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         pan_delete_1_s.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  *************************************************************************
--  *
--  *    Panel_delete_1                        -- Package SPEC
--  *
--  *************************************************************************

with TAE;
with X_Windows;

package Panel_delete_1 is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  delete_1
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
```

```
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  delete_1
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;   -- panel information


  --  ......................................................................
  --  .
  --  .    Initialize_Panel                   -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Initialize_Panel
    ( Collection_Read                        -- TAE Collection read from
        : in TAE.Tae_Co.Collection_Ptr );    -- resource file

  --| PURPOSE:
  --| This procedure initializes the Info.Target and Info.View for this panel
  --|
  --| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
  --| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
  --|    Collection_Read
  --|
  --| NOTES: (none)





  --  ......................................................................
  --  .
  --  .    Create_Panel                       -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Create_Panel
    ( Panel_State                            -- Flags sent to Wpt_NewPanel.
        : in TAE.Tae_Wpt.Wpt_Flags
          := TAE.Tae_Wpt.WPT_PREFERRED;

      Relative_Window                        -- Panel origin is offset from
        : in X_Windows.Window                -- this X Window.  Null_Window
```

```
         := X_Windows.Null_Window );        -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--  .......................................................................
--  .
--  .    Connect_Panel                      -- Subprogram SPEC
--  .
--  .......................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                         -- Panel origin is offset from
      : in X_Windows.Window                 -- this X Window.  Null_Window
        := X_Windows.Null_Window );         -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--  .......................................................................
--  .
--  .    Destroy_Panel                      -- Subprogram SPEC
--  .
--  .......................................................................
```

```
   procedure Destroy_Panel;

   --| PURPOSE:
   --| This procedure erases a panel from the screen and de-allocates the
   --| associated panel object (not the target and view).
   --|
   --| EXCEPTIONS:
   --| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
   --|
   --| NOTES:
   --| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
   --| in any Wpt call until it is created again.




   --  ....................................................................
   --  .
   --  .    Dispatch_Item                     -- Subprogram SPEC
   --  .
   --  ....................................................................

   procedure Dispatch_Item
     ( User_Context_Ptr                       -- Wpt Event Context for a PARM
         : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

   --| PURPOSE:
   --| This procedure calls the Event Handler specified by User_Context_Ptr
   --|
   --| EXCEPTIONS:
   --| Application-specific
   --|
   --| NOTES: (none)

end Panel_delete_1;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_delete_1_b.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author     : Ruey-Wen (Vincent) Hong
-- *** Date       : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ***************************************************************************
--  *
--  *    Panel_delete_1                    -- Package BODY
--  *
--  ***************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.
-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_delete_1 is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  delete_1
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   delete_1_disp
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

  --  .....................................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram BODY
  --  .
  --  .....................................................................

  procedure Initialize_Panel
    ( Collection_Read
      : in TAE.Tae_Co.Collection_Ptr ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

begin -- Initialize_Panel

  Info := new TAE.Tae_Wpt.Event_Context;
  Info.Collection := Collection_Read;
  TAE.Tae_Co.Co_Find (Info.Collection, "delete_1_v", Info.View);
  TAE.Tae_Co.Co_Find (Info.Collection, "delete_1_t", Info.Target);

  -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_delete_1.Initialize_Panel:  "
      & "Collection_Read not initialized.");
    raise;

  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_delete_1.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  ...............................................................
--  .
--  .    Create_Panel                    -- Subprogram BODY
--  .
--  ...............................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel
```

```
  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id             => Global.Default_Display_Id,
        Data_Vm                => Info.Target,
        View_Vm                => Info.View,
        Relative_Window        => Relative_Window,
        User_Context           => Info,
        Flags                  => Panel_State,
        Panel_Id               => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (delete_1) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_delete_1.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_delete_1.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  ...................................................................
--  .
--  .    Connect_Panel                       -- Subprogram BODY
--  .
--  ...................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel
```

```
  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window       => Relative_Window,
        Panel_State           => Panel_State );
  else
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_STATE =>
    Text_IO.Put_Line ("Panel_delete_1.Connect_Panel:  "
      & "Invalid panel state.");
    raise;

end Connect_Panel;



--  ...................................................................
--  .
--  .    Destroy_Panel                      -- Subprogram BODY
--  .
--  ...................................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_delete_1.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;
```

```
end Destroy_Panel;



--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--  ......................................................................
--  .
--  .    delete_1_disp_Event             -- Subprogram SPEC & BODY
--  .
--  ......................................................................

procedure delete_1_disp_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: delete_1_disp.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: delete_1_disp.

begin -- delete_1_disp_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel delete_1, parm delete_1_disp: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: delete_1_disp.
  -- MERGE NOTE: Add code ABOVE this line for parm: delete_1_disp.

end delete_1_disp_Event;



--
-- end EVENT HANDLERS
--
--------------------------------------------------------------------------
```

```
  --  ...................................................................
  --  .
  --  .    Dispatch_Item                       -- Subprogram BODY
  --  .
  --  ...................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| NOTES: (none)

  begin -- Dispatch_Item

    if TAE.Tae_Misc.s_equal ("delete_l_disp", User_Context_Ptr.Parm_Name) then
      delete_l_disp_Event (User_Context_Ptr);
    end if;

  end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_delete_l;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_go_s.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author     : Ruey-Wen (Vincent) Hong
-- *** Date       : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ***************************************************************************
--  *
--  *    Panel_go                            -- Package SPEC
--  *
--  ***************************************************************************

with TAE;
with X_Windows;

package Panel_go is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  go
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
```

```
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  go
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;    -- panel information


  --  ......................................................................
  --  .
  --  .    Initialize_Panel                   -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Initialize_Panel
    ( Collection_Read                         -- TAE Collection read from
        : in TAE.Tae_Co.Collection_Ptr );     -- resource file

  --| PURPOSE:
  --| This procedure initializes the Info.Target and Info.View for this panel
  --|
  --| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
  --| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
  --|    Collection_Read
  --|
  --| NOTES: (none)





  --  ......................................................................
  --  .
  --  .    Create_Panel                       -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Create_Panel
    ( Panel_State                             -- Flags sent to Wpt_NewPanel.
        : in TAE.Tae_Wpt.Wpt_Flags
          := TAE.Tae_Wpt.WPT_PREFERRED;

      Relative_Window                         -- Panel origin is offset from
        : in X_Windows.Window                 -- this X Window.  Null_Window
          := X_Windows.Null_Window );         -- uses the root window.

  --| PURPOSE:
  --| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--  .......................................................................
--  .
--  .    Connect_Panel                  -- Subprogram SPEC
--  .
--  .......................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                     -- Panel origin is offset from
      : in X_Windows.Window             -- this X Window.  Null_Window
        := X_Windows.Null_Window );     -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--  .......................................................................
--  .
--  .    Destroy_Panel                  -- Subprogram SPEC
--  .
--  .......................................................................

procedure Destroy_Panel;

--| PURPOSE:
--| This procedure erases a panel from the screen and de-allocates the
```

```
  --| associated panel object (not the target and view).
  --|
  --| EXCEPTIONS:
  --| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
  --|
  --| NOTES:
  --| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
  --| in any Wpt call until it is created again.




  --  .......................................................................
  --  .
  --  .    Dispatch_Item                       -- Subprogram SPEC
  --  .
  --  .......................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr                         -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_go;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         pan_go_b.a
-- *** Generated:    Feb 22 12:17:44 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ***************************************************************************
--  *
--  *    Panel_go                              -- Package BODY
--  *
--  ***************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.
with Panel_grnd_eq;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.
```

```
-----------------------------------------------------------------
-- ADDED By RUEY-WEN HONG 960222
-----------------------------------------------------------------
With Op_Util_Mods; Use Op_Util_Mods;
With Global_def; Use Global_def;
With Query_processing_pkg; Use Query_processing_pkg;
With Swb_pkg; Use Swb_pkg;
With Swb_def_pkg; Use Swb_def_pkg;
With Signature_match_pkg; Use Signature_match_pkg;
With Semantic_match_pkg; Use Semantic_match_pkg;
With Init_pkg; Use Init_pkg;
With Formulate_Result_Output_pkg; Use Formulate_Result_Output_pkg;
With Float_io; Use Float_io;
With Integer_io; Use Integer_io;
With Text_io; Use Text_io;
With Integer_io; Use Integer_io;
-----------------------------------------------------------------

package body Panel_go is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  go
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   Cancel,           go_help,         go_label,        go_search,
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--| 22-Feb-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

-----------------------------------------------------------------
-- ADDED By RUEY-WEN HONG 960222
-----------------------------------------------------------------
  Candidates : CandidatesTable;
  Profile_err : Profile_err_list := (Others => Nill);
  Q: Qc; C: SWC;
  V: Signature_map;
  id,i : Natural := 1;
  c1 : natural := 0;
  sn : natural;
  Num_vmaps : Natural;
  Stable : Sort_table_def;

  Ul,Ll : Float;

  KeywordRank, ProfileRank : Float;
  SortArray : Asort_Array;
-----------------------------------------------------------------

  --  ..............................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram BODY
  --  .
  --  ..............................................................

  procedure Initialize_Panel
    ( Collection_Read
        : in TAE.Tae_Co.Collection_Ptr ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

  begin -- Initialize_Panel

    Info := new TAE.Tae_Wpt.Event_Context;
    Info.Collection := Collection_Read;
    TAE.Tae_Co.Co_Find (Info.Collection, "go_v", Info.View);
    TAE.Tae_Co.Co_Find (Info.Collection, "go_t", Info.Target);

    -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

  exception

    when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_go.Initialize_Panel:  "
      & "Collection_Read not initialized.");
    raise;

  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_go.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  .....................................................................
--  .
--  .   Create_Panel                      -- Subprogram BODY
--  .
--  .....................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id            => Global.Default_Display_Id,
        Data_Vm               => Info.Target,
        View_Vm               => Info.View,
        Relative_Window       => Relative_Window,
        User_Context          => Info,
        Flags                 => Panel_State,
        Panel_Id              => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (go) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
```

```
      Text_IO.Put_Line ("Panel_go.Create_Panel:  "
        & "Panel was not initialized prior to creation.");
      raise;

    when TAE.TAE_FAIL =>
      Text_IO.Put_Line ("Panel_go.Create_Panel:  "
        & "Panel could not be created.");
      raise;

  end Create_Panel;



  --  ....................................................................
  --  .
  --  .    Connect_Panel                        -- Subprogram BODY
  --  .
  --  ....................................................................

  procedure Connect_Panel
    ( Panel_State
        : in TAE.Tae_Wpt.Wpt_Flags
          := TAE.Tae_Wpt.WPT_PREFERRED;

      Relative_Window
        : in X_Windows.Window
          := X_Windows.Null_Window ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

  begin -- Connect_Panel

    if Info.Panel_Id = Tae.Null_Panel_Id then
      Create_Panel
        ( Relative_Window       => Relative_Window,
          Panel_State           => Panel_State );
    else
      TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
    end if;

    -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
    -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

  exception

    when TAE.Tae_Wpt.BAD_STATE =>
      Text_IO.Put_Line ("Panel_go.Connect_Panel:  "
        & "Invalid panel state.");
      raise;

  end Connect_Panel;
```

```
--  ....................................................................
--  .
--  .    Destroy_Panel                        -- Subprogram BODY
--  .
--  ....................................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_go.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;


--++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--  ....................................................................
--  .
--  .    Cancel_Event                         -- Subprogram SPEC & BODY
--  .
--  ....................................................................

procedure Cancel_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
```

```
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: Cancel.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: Cancel.

begin -- Cancel_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel go, parm Cancel: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- Cancel
  Destroy_Panel;
  Panel_grnd_eq.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: Cancel.
  -- MERGE NOTE: Add code ABOVE this line for parm: Cancel.

end Cancel_Event;


--  .....................................................................
--  .
--  .    go_help_Event                    -- Subprogram SPEC & BODY
--  .
--  .....................................................................

procedure go_help_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: go_help.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: go_help.

begin -- go_help_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
```

```
    Text_IO.Put ("Panel go, parm go_help: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: go_help.
    -- MERGE NOTE: Add code ABOVE this line for parm: go_help.

  end go_help_Event;



  --  ....................................................................
  --  .
  --  .    go_label_Event                    -- Subprogram SPEC & BODY
  --  .
  --  ....................................................................

  procedure go_label_Event
    ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| PURPOSE:
  --| EVENT HANDLER.  Insert application specific information.
  --|
  --| NOTES: (none)

    Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
    Count : TAE.Taeint;

    -- MERGE NOTE: Add declarations BELOW this line for parm: go_label.
    -- MERGE NOTE: Add declarations ABOVE this line for parm: go_label.

  begin -- go_label_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel go, parm go_label: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: go_label.
    -- MERGE NOTE: Add code ABOVE this line for parm: go_label.

  end go_label_Event;



  --  ....................................................................
```

```
--  .
--  .    go_search_Event                     -- Subprogram SPEC & BODY
--  .
--  .......................................................................

procedure go_search_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

    Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
    Count : TAE.Taeint;

    -- MERGE NOTE: Add declarations BELOW this line for parm: go_search.
    -- MERGE NOTE: Add declarations ABOVE this line for parm: go_search.

begin -- go_search_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel go, parm go_search: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: go_search.
    -- MERGE NOTE: Add code ABOVE this line for parm: go_search.


-----------------------------------------------------------------
-- ADDED By RUEY-WEN HONG 960222
-----------------------------------------------------------------

--     put("Enter Ll,Ul: ");
--     get(Ll); get(Ul);

--     Put("Enter Sn: ");
--     get(sn);
--     get(id);

--       Ll := 0.1;

       Ll:= Global.kps_from;
       Ul:= Global.kps_to;
       sn := Global.mbn;
       id := 1;


    Get_Query(Q,Stable);
    -- Retrieve components from Library using profile from test cases

        Initialize_variables(V,Num_vmaps,Stable);

        InitComponentData(id,C,Stable,Q,KeywordRank,ProfileRank);

        -- Signature Match

        put("Working on :");put(String(C.Obj_filename));New_line;

        c1 := 0;

Signature_Match(Q,C,V,Stable,Num_vmaps,c1,sn,L1,U1,KeywordRank,ProfileRank);

        put("Count number of unfilter map"); put(c1);new_line;

        -- Semantic_Match
        Semantic_Match(Q,C,V,Stable,Num_vmaps);

        -- Formulate Result and output to user
        Calculate_total_rank(V,Q,C,i,Num_vmaps,SortArray,
        KeywordRank,ProfileRank);


    Sort_Display_Result(SortArray,1);

    Display_invalid_operations(Q,Profile_err);
-----------------------------------------------------------------


  end go_search_Event;



  --
  -- end EVENT HANDLERS
  --
  --------------------------------------------------------------------------



  --  ......................................................................
  --  .
  --  .    Dispatch_Item                     -- Subprogram BODY
  --  .
  --  ......................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| NOTES: (none)

  begin -- Dispatch_Item
```

```
    if TAE.Tae_Misc.s_equal ("Cancel", User_Context_Ptr.Parm_Name) then
      Cancel_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("go_help", User_Context_Ptr.Parm_Name) then
      go_help_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("go_label", User_Context_Ptr.Parm_Name) then
      go_label_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("go_search", User_Context_Ptr.Parm_Name) then
      go_search_Event (User_Context_Ptr);
    end if;

  end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_go;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_grnd_eq_s.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  *************************************************************************
--  *
--  *    Panel_grnd_eq                       -- Package SPEC
--  *
--  *************************************************************************

with TAE;
with X_Windows;

package Panel_grnd_eq is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  grnd_eq
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  grnd_eq
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;   -- panel information


  --  ......................................................................
  --  .
  --  .    Initialize_Panel                   -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Initialize_Panel
    ( Collection_Read                       -- TAE Collection read from
        : in TAE.Tae_Co.Collection_Ptr );   -- resource file

  --| PURPOSE:
  --| This procedure initializes the Info.Target and Info.View for this panel
  --|
  --| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
  --| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
  --|    Collection_Read
  --|
  --| NOTES: (none)




  --  ......................................................................
  --  .
  --  .    Create_Panel                       -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Create_Panel
    ( Panel_State                           -- Flags sent to Wpt_NewPanel.
        : in TAE.Tae_Wpt.Wpt_Flags
          := TAE.Tae_Wpt.WPT_PREFERRED;

      Relative_Window                       -- Panel origin is offset from
        : in X_Windows.Window               -- this X Window.  Null_Window
          := X_Windows.Null_Window );       -- uses the root window.

  --| PURPOSE:
  --| This procedure creates this panel object in the specified Panel_State
  --| and stores the panel Id in Info.Panel_Id.
  --|
  --| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
  --| TAE.TAE_FAIL is raised if the panel could not be created
  --|
--| NOTES: (none)



--  ............................................................
--  .
--  .    Connect_Panel                     -- Subprogram SPEC
--  .
--  ............................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                       -- Panel origin is offset from
      : in X_Windows.Window               -- this X Window.  Null_Window
        := X_Windows.Null_Window );       -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--  ............................................................
--  .
--  .    Destroy_Panel                     -- Subprogram SPEC
--  .
--  ............................................................

procedure Destroy_Panel;

--| PURPOSE:
--| This procedure erases a panel from the screen and de-allocates the
--| associated panel object (not the target and view).
--|
--| EXCEPTIONS:
--| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
--|
--| NOTES:
  --| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
  --| in any Wpt call until it is created again.




  -- ..........................................................................
  -- .
  -- .    Dispatch_Item                       -- Subprogram SPEC
  -- .
  -- ..........................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr                        -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_grnd_eq;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_grnd_eq_b.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ***************************************************************************
--  *
--  *    Panel_grnd_eq                        -- Package BODY
--  *
--  ***************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.
with Panel_mbn_kps;
with Panel_grnd_eq;
with Panel_go;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_grnd_eq is

--| NOTES:
```

```
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  grnd_eq
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   grnd_eq_cancel,  grnd_eq_help,    grnd_eq_keyin,   grnd_eq_next,
--|   grnd_eq_ok
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.


  --  .................................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram BODY
  --  .
  --  .................................................................

  procedure Initialize_Panel
    ( Collection_Read
        : in TAE.Tae_Co.Collection_Ptr ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.
```

```
begin -- Initialize_Panel

  Info := new TAE.Tae_Wpt.Event_Context;
  Info.Collection := Collection_Read;
  TAE.Tae_Co.Co_Find (Info.Collection, "grnd_eq_v", Info.View);
  TAE.Tae_Co.Co_Find (Info.Collection, "grnd_eq_t", Info.Target);

  -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_grnd_eq.Initialize_Panel:  "
      & "Collection_Read not initialized.");
    raise;

  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_grnd_eq.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  .......................................................................
--  .
--  .    Create_Panel                        -- Subprogram BODY
--  .
--  .......................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id          => Global.Default_Display_Id,
        Data_Vm             => Info.Target,
        View_Vm             => Info.View,
        Relative_Window        => Relative_Window,
        User_Context           => Info,
        Flags                  => Panel_State,
        Panel_Id               => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (grnd_eq) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_grnd_eq.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_grnd_eq.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  .....................................................................
--  .
--  .    Connect_Panel                      -- Subprogram BODY
--  .
--  .....................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window        => Relative_Window,
        Panel_State            => Panel_State );
  else
```

```
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_STATE =>
    Text_IO.Put_Line ("Panel_grnd_eq.Connect_Panel:  "
      & "Invalid panel state.");
    raise;

end Connect_Panel;



--  ......................................................................
--  .
--  .    Destroy_Panel                       -- Subprogram BODY
--  .
--  ......................................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_grnd_eq.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;


--++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--  ...........................................................
--  .
--  .    grnd_eq_cancel_Event             -- Subprogram SPEC & BODY
--  .
--  ...........................................................

procedure grnd_eq_cancel_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: grnd_eq_cancel.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: grnd_eq_cancel.

begin -- grnd_eq_cancel_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel grnd_eq, parm grnd_eq_cancel: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- grnd_eq_cancel
  Destroy_Panel;
  Panel_mbn_kps.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: grnd_eq_cancel.
  -- MERGE NOTE: Add code ABOVE this line for parm: grnd_eq_cancel.

end grnd_eq_cancel_Event;



--  ...........................................................
--  .
--  .    grnd_eq_help_Event               -- Subprogram SPEC & BODY
--  .
--  ...........................................................

procedure grnd_eq_help_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is
```

```
--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: grnd_eq_help.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: grnd_eq_help.

begin -- grnd_eq_help_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel grnd_eq, parm grnd_eq_help: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- grnd_eq_help
  Destroy_Panel;
  Panel_mbn_kps.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: grnd_eq_help.
  -- MERGE NOTE: Add code ABOVE this line for parm: grnd_eq_help.

end grnd_eq_help_Event;



--  ...............................................................
--  .
--  .    grnd_eq_keyin_Event               -- Subprogram SPEC & BODY
--  .
--  ...............................................................

procedure grnd_eq_keyin_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: grnd_eq_keyin.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: grnd_eq_keyin.

begin -- grnd_eq_keyin_Event
```

```
  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel grnd_eq, parm grnd_eq_keyin: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: grnd_eq_keyin.
  -- MERGE NOTE: Add code ABOVE this line for parm: grnd_eq_keyin.

end grnd_eq_keyin_Event;



--  ..................................................................
--  .
--  .    grnd_eq_next_Event               -- Subprogram SPEC & BODY
--  .
--  ..................................................................

procedure grnd_eq_next_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: grnd_eq_next.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: grnd_eq_next.

begin -- grnd_eq_next_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel grnd_eq, parm grnd_eq_next: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- grnd_eq_next
  Destroy_Panel;
  Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: grnd_eq_next.
  -- MERGE NOTE: Add code ABOVE this line for parm: grnd_eq_next.
```

```
end grnd_eq_next_Event;



--  ............................................................
--  .
--  .    grnd_eq_ok_Event                  -- Subprogram SPEC & BODY
--  .
--  ............................................................

procedure grnd_eq_ok_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: grnd_eq_ok.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: grnd_eq_ok.

begin -- grnd_eq_ok_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel grnd_eq, parm grnd_eq_ok: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- grnd_eq_ok
  Destroy_Panel;
  Panel_go.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: grnd_eq_ok.
  -- MERGE NOTE: Add code ABOVE this line for parm: grnd_eq_ok.

end grnd_eq_ok_Event;



--
-- end EVENT HANDLERS
--
-------------------------------------------------------------------------
```

```
  -- ..................................................................
  -- .
  -- .    Dispatch_Item                       -- Subprogram BODY
  -- .
  -- ..................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| NOTES: (none)

  begin -- Dispatch_Item

    if TAE.Tae_Misc.s_equal ("grnd_eq_cancel", User_Context_Ptr.Parm_Name)
then
      grnd_eq_cancel_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("grnd_eq_help", User_Context_Ptr.Parm_Name)
then
      grnd_eq_help_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("grnd_eq_keyin", User_Context_Ptr.Parm_Name)
then
      grnd_eq_keyin_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("grnd_eq_next", User_Context_Ptr.Parm_Name)
then
      grnd_eq_next_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("grnd_eq_ok", User_Context_Ptr.Parm_Name) then
      grnd_eq_ok_Event (User_Context_Ptr);
    end if;

  end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_grnd_eq;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         pan_keyword_s.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  **************************************************************************
--  *
--  *    Panel_keyword                        -- Package SPEC
--  *
--  **************************************************************************

with TAE;
with X_Windows;

package Panel_keyword is
```

```
--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  keyword
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  keyword
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;   -- panel information


  --  ......................................................................
  --  .
  --  .    Initialize_Panel                -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Initialize_Panel
    ( Collection_Read                     -- TAE Collection read from
        : in TAE.Tae_Co.Collection_Ptr );   -- resource file

  --| PURPOSE:
  --| This procedure initializes the Info.Target and Info.View for this panel
  --|
  --| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
  --| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
  --|    Collection_Read
  --|
  --| NOTES: (none)




  --  ......................................................................
  --  .
  --  .    Create_Panel                    -- Subprogram SPEC
  --  .
  --  ......................................................................
```

```
procedure Create_Panel
  ( Panel_State                              -- Flags sent to Wpt_NewPanel.
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                          -- Panel origin is offset from
      : in X_Windows.Window                  -- this X Window.  Null_Window
        := X_Windows.Null_Window );          -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--  ....................................................................
--  .
--  .    Connect_Panel                       -- Subprogram SPEC
--  .
--  ....................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                          -- Panel origin is offset from
      : in X_Windows.Window                  -- this X Window.  Null_Window
        := X_Windows.Null_Window );          -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)
```

```
  --  ............................................................
  --  .
  --  .    Destroy_Panel                    -- Subprogram SPEC
  --  .
  --  ............................................................

  procedure Destroy_Panel;

  --| PURPOSE:
  --| This procedure erases a panel from the screen and de-allocates the
  --| associated panel object (not the target and view).
  --|
  --| EXCEPTIONS:
  --| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
  --|
  --| NOTES:
  --| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
  --| in any Wpt call until it is created again.




  --  ............................................................
  --  .
  --  .    Dispatch_Item                    -- Subprogram SPEC
  --  .
  --  ............................................................

  procedure Dispatch_Item
    ( User_Context_Ptr                      -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_keyword;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         pan_keyword_b.a
-- *** Generated:    Feb 20 14:23:54 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ***************************************************************************
--  *
--  *    Panel_keyword                          -- Package BODY
--  *
--  ****************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

use Text_IO,Global;

-- One "with" statement for each connected panel.
with Panel_query;
with Panel_op_no;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_keyword is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  keyword
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   keyword_cancel,  keyword_help,    keyword_list,    keyword_ok,
--|   selected_kword
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--| 20-Feb-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.
```

```
 -- MERGE NOTE: Add additional code BELOW this line.
 -- MERGE NOTE: Add additional code ABOVE this line.

--ADDED by Ruey_Wen (Vincent) Hong 951128


 kw_vec        : s_vector(1..100):= (others=> new STRING(1..80));
 kwadd_vec     : s_vector(1..15) := (others=> new STRING(1..80));
 kw_selected   : String (1..80)  := (others=>' ');
 kw_list       : file_type;
 file_name     : String(1..7) := "kw_list";


 number,
 index        : integer := 1;

 Type_of_Wpt_Event       : Tae.Wpt_Eventtype;
 UNKNOWN_WPT_EVENT       : Exception;


 --  ..................................................................
 --  .
 --  .    Initialize_Panel                  -- Subprogram BODY
 --  .
 --  ..................................................................

 procedure Initialize_Panel
   ( Collection_Read
       : in TAE.Tae_Co.Collection_Ptr ) is

 --| NOTES: (none)

   -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
   -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

 begin -- Initialize_Panel

   Info := new TAE.Tae_Wpt.Event_Context;
   Info.Collection := Collection_Read;
   TAE.Tae_Co.Co_Find (Info.Collection, "keyword_v", Info.View);
   TAE.Tae_Co.Co_Find (Info.Collection, "keyword_t", Info.Target);

   -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
   -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

 exception

   when TAE.UNINITIALIZED_PTR =>
     Text_IO.Put_Line ("Panel_keyword.Initialize_Panel:  "
       & "Collection_Read not initialized.");
     raise;
```

```
  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_keyword.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  ...................................................................
--  .
--  .    Create_Panel                         -- Subprogram BODY
--  .
--  ...................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id              => Global.Default_Display_Id,
        Data_Vm                 => Info.Target,
        View_Vm                 => Info.View,
        Relative_Window         => Relative_Window,
        User_Context            => Info,
        Flags                   => Panel_State,
        Panel_Id                => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (keyword) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_keyword.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;
    when TAE.TAE_FAIL =>
      Text_IO.Put_Line ("Panel_keyword.Create_Panel:  "
        & "Panel could not be created.");
      raise;

  end Create_Panel;



  --  .......................................................................
  --  .
  --  .    Connect_Panel                        -- Subprogram BODY
  --  .
  --  .......................................................................

  procedure Connect_Panel
    ( Panel_State
        : in TAE.Tae_Wpt.Wpt_Flags
          := TAE.Tae_Wpt.WPT_PREFERRED;

      Relative_Window
        : in X_Windows.Window
          := X_Windows.Null_Window ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

 --ADDED

   Dummy:Boolean;
   L    :integer:=1;

  begin -- Connect_Panel

    if Info.Panel_Id = Tae.Null_Panel_Id then
      Create_Panel
        ( Relative_Window       => Relative_Window,
          Panel_State           => Panel_State );
    else
      TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
    end if;

    -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
    -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

-----------------------------------------------------------------------
  -- ADDED
-----------------------------------------------------------------------

Global.strlen(library,L);
Global.list_components(kw_list,file_name,kw_vec,number);
```

```
  TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);

TAE.Tae_Wpt.Wpt_SetStringConstraints(Info.Panel_Id,"keyword_list",taeint(numb
er),kw_vec);


  Dummy:=TAE.Tae_Wpt.Wpt_Pending;

--  system_call("rm kw_list");


  exception

    when TAE.Tae_Wpt.BAD_STATE =>
      Text_IO.Put_Line ("Panel_keyword.Connect_Panel:  "
        & "Invalid panel state.");
      raise;

  end Connect_Panel;



  --  ...................................................................
  --  .
  --  .    Destroy_Panel                    -- Subprogram BODY
  --  .
  --  ...................................................................

  procedure Destroy_Panel is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

  begin -- Destroy_Panel

    TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

    -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
    -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

  exception

    when TAE.Tae_Wpt.BAD_PANEL_ID =>
      Text_IO.Put_Line ("Panel_keyword.Destroy_Panel:  "
        & "Info.Panel_Id is an invalid id.");
      raise;

    when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
      -- This panel has not been created yet, or has already been destroyed.
      -- Trap this exception and do nothing.
      null;
```

```
end Destroy_Panel;



--++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--   ...................................................................
--   .
--   .    keyword_cancel_Event             -- Subprogram SPEC & BODY
--   .
--   ...................................................................

procedure keyword_cancel_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: keyword_cancel.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: keyword_cancel.

begin -- keyword_cancel_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel keyword, parm keyword_cancel: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- keyword_cancel
  Destroy_Panel;
  Panel_query.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: keyword_cancel.
  -- MERGE NOTE: Add code ABOVE this line for parm: keyword_cancel.

end keyword_cancel_Event;



--   ...................................................................
--   .
--   .    keyword_help_Event               -- Subprogram SPEC & BODY
--   .
-- ..........................................................................

procedure keyword_help_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: keyword_help.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: keyword_help.

begin -- keyword_help_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel keyword, parm keyword_help: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- keyword_help
  Destroy_Panel;
  Panel_query.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: keyword_help.
  -- MERGE NOTE: Add code ABOVE this line for parm: keyword_help.

end keyword_help_Event;



-- ..........................................................................
--  .
--  .    keyword_list_Event                -- Subprogram SPEC & BODY
--  .
-- ..........................................................................

procedure keyword_list_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;
```

```
    -- MERGE NOTE: Add declarations BELOW this line for parm: keyword_list.
    -- MERGE NOTE: Add declarations ABOVE this line for parm: keyword_list.

  begin -- keyword_list_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel keyword, parm keyword_list: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: keyword_list.
    -- MERGE NOTE: Add code ABOVE this line for parm: keyword_list.


 ---------------------------------------------------------------
 --ADDED
 ---------------------------------------------------------------

        kw_selected(1..80):=Value(1)(1..80);
         kwadd_vec(index).all:=kw_selected;


TAE.Tae_Wpt.Wpt_SetStringConstraints(Info.Panel_Id,"selected_kword",taeint(in
dex),kwadd_vec);

         index := index + 1;

  end keyword_list_Event;



  --   ....................................................................
  --   .
  --   .    keyword_ok_Event                  -- Subprogram SPEC & BODY
  --   .
  --   ....................................................................

  procedure keyword_ok_Event
    ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| PURPOSE:
  --| EVENT HANDLER.  Insert application specific information.
  --|
  --| NOTES: (none)

    Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
    Count : TAE.Taeint;

    -- MERGE NOTE: Add declarations BELOW this line for parm: keyword_ok.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: keyword_ok.

begin -- keyword_ok_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel keyword, parm keyword_ok: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- keyword_ok
  Destroy_Panel;
  Panel_op_no.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: keyword_ok.
  -- MERGE NOTE: Add code ABOVE this line for parm: keyword_ok.

end keyword_ok_Event;



--  .....................................................................
--  .
--  .    selected_kword_Event              -- Subprogram SPEC & BODY
--  .
--  .....................................................................

procedure selected_kword_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: selected_kword.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: selected_kword.

begin -- selected_kword_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel keyword, parm selected_kword: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;
```

```
    -- MERGE NOTE: Add code BELOW this line for parm: selected_kword.
    -- MERGE NOTE: Add code ABOVE this line for parm: selected_kword.

  end selected_kword_Event;



  --
  -- end EVENT HANDLERS
  --
  -------------------------------------------------------------------------



  --  ......................................................................
  --  .
  --  .    Dispatch_Item                        -- Subprogram BODY
  --  .
  --  ......................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| NOTES: (none)

  begin -- Dispatch_Item

    if TAE.Tae_Misc.s_equal ("keyword_cancel", User_Context_Ptr.Parm_Name)
then
      keyword_cancel_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("keyword_help", User_Context_Ptr.Parm_Name)
then
      keyword_help_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("keyword_list", User_Context_Ptr.Parm_Name)
then
      keyword_list_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("keyword_ok", User_Context_Ptr.Parm_Name) then
      keyword_ok_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("selected_kword",
User_Context_Ptr.Parm_Name) then
      selected_kword_Event (User_Context_Ptr);
    end if;

  end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_keyword;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_maintain_s.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ************************************************************************
--  *
--  *    Panel_maintain                          -- Package SPEC
--  *
--  ************************************************************************

with TAE;
with X_Windows;

package Panel_maintain is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  maintain
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  maintain
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;    -- panel information


  --  ......................................................................
  --  .
  --  .    Initialize_Panel                     -- Subprogram SPEC
  --  .
  --  ......................................................................

  procedure Initialize_Panel
    ( Collection_Read                         -- TAE Collection read from
        : in TAE.Tae_Co.Collection_Ptr );     -- resource file

  --| PURPOSE:
  --| This procedure initializes the Info.Target and Info.View for this panel
  --|
  --| EXCEPTIONS:
```

```
--| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
--| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
--|    Collection_Read
--|
--| NOTES: (none)




--  ......................................................................
--  .
--  .    Create_Panel                    -- Subprogram SPEC
--  .
--  ......................................................................

procedure Create_Panel
  ( Panel_State                         -- Flags sent to Wpt_NewPanel.
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                     -- Panel origin is offset from
      : in X_Windows.Window             -- this X Window.  Null_Window
        := X_Windows.Null_Window );     -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--  ......................................................................
--  .
--  .    Connect_Panel                   -- Subprogram SPEC
--  .
--  ......................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                     -- Panel origin is offset from
      : in X_Windows.Window             -- this X Window.  Null_Window
        := X_Windows.Null_Window );     -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--  ...............................................................
--  .
--  .    Destroy_Panel                      -- Subprogram SPEC
--  .
--  ...............................................................

procedure Destroy_Panel;

--| PURPOSE:
--| This procedure erases a panel from the screen and de-allocates the
--| associated panel object (not the target and view).
--|
--| EXCEPTIONS:
--| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
--|
--| NOTES:
--| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
--| in any Wpt call until it is created again.




--  ...............................................................
--  .
--  .    Dispatch_Item                      -- Subprogram SPEC
--  .
--  ...............................................................

procedure Dispatch_Item
  ( User_Context_Ptr                    -- Wpt Event Context for a PARM
      : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

--| PURPOSE:
--| This procedure calls the Event Handler specified by User_Context_Ptr
--|
--| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_maintain;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         pan_maintain_b.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ************************************************************************
--  *
--  *    Panel_maintain                      -- Package BODY
--  *
--  ************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.
with Panel_backup_l;
with Panel_sb_main;
with Panel_create_l;
with Panel_delete_l;
with Panel_select_l;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_maintain is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  maintain
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   maint_backup,    maint_cancel,    maint_create,    maint_delete,
--|   maint_help,      maint_select
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

  --  .......................................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram BODY
  --  .
  --  .......................................................................

  procedure Initialize_Panel
    ( Collection_Read
        : in TAE.Tae_Co.Collection_Ptr ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

  begin -- Initialize_Panel

    Info := new TAE.Tae_Wpt.Event_Context;
    Info.Collection := Collection_Read;
    TAE.Tae_Co.Co_Find (Info.Collection, "maintain_v", Info.View);
    TAE.Tae_Co.Co_Find (Info.Collection, "maintain_t", Info.Target);

    -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

  exception

    when TAE.UNINITIALIZED_PTR =>
      Text_IO.Put_Line ("Panel_maintain.Initialize_Panel:  "
        & "Collection_Read not initialized.");
      raise;

    when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_maintain.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  .........................................................................
--  .
--  .    Create_Panel                          -- Subprogram BODY
--  .
--  .........................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id            => Global.Default_Display_Id,
        Data_Vm               => Info.Target,
        View_Vm               => Info.View,
        Relative_Window       => Relative_Window,
        User_Context          => Info,
        Flags                 => Panel_State,
        Panel_Id              => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (maintain) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_maintain.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
```

```
    Text_IO.Put_Line ("Panel_maintain.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  ...................................................................
--  .
--  .    Connect_Panel                    -- Subprogram BODY
--  .
--  ...................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window       => Relative_Window,
        Panel_State           => Panel_State );
  else
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_STATE =>
    Text_IO.Put_Line ("Panel_maintain.Connect_Panel:  "
      & "Invalid panel state.");
    raise;

end Connect_Panel;



--  ...................................................................
--  .
--  .    Destroy_Panel                          -- Subprogram BODY
--  .
--  ..........................................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_maintain.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;



--++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--  ..........................................................................
--  .
--  .    maint_backup_Event                     -- Subprogram SPEC & BODY
--  .
--  ..........................................................................

procedure maint_backup_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;
  -- MERGE NOTE: Add declarations BELOW this line for parm: maint_backup.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: maint_backup.

begin -- maint_backup_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel maintain, parm maint_backup: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- maint_backup
  Destroy_Panel;
  Panel_backup_1.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: maint_backup.
  -- MERGE NOTE: Add code ABOVE this line for parm: maint_backup.

end maint_backup_Event;



--  ...........................................................
--  .
--  .    maint_cancel_Event                -- Subprogram SPEC & BODY
--  .
--  ...........................................................

procedure maint_cancel_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: maint_cancel.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: maint_cancel.

begin -- maint_cancel_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel maintain, parm maint_cancel: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
```

```
    Text_IO.Put_Line ("none");
  end if;

  -- maint_cancel
  Destroy_Panel;
  Panel_sb_main.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: maint_cancel.
  -- MERGE NOTE: Add code ABOVE this line for parm: maint_cancel.

end maint_cancel_Event;



--  ..................................................................
--  .
--  .    maint_create_Event              -- Subprogram SPEC & BODY
--  .
--  ..................................................................

procedure maint_create_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: maint_create.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: maint_create.

begin -- maint_create_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel maintain, parm maint_create: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- maint_create
  Destroy_Panel;
  Panel_create_1.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: maint_create.
  -- MERGE NOTE: Add code ABOVE this line for parm: maint_create.

end maint_create_Event;
```

```
--  .............................................................
--  .
--  .    maint_delete_Event                -- Subprogram SPEC & BODY
--  .
--  .............................................................

procedure maint_delete_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: maint_delete.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: maint_delete.

begin -- maint_delete_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel maintain, parm maint_delete: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- maint_delete
  Destroy_Panel;
  Panel_delete_1.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: maint_delete.
  -- MERGE NOTE: Add code ABOVE this line for parm: maint_delete.

end maint_delete_Event;



--  .............................................................
--  .
--  .    maint_help_Event                  -- Subprogram SPEC & BODY
--  .
--  .............................................................

procedure maint_help_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
```

```
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: maint_help.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: maint_help.

begin -- maint_help_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel maintain, parm maint_help: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: maint_help.
  -- MERGE NOTE: Add code ABOVE this line for parm: maint_help.

end maint_help_Event;



--  .....................................................................
--  .
--  .    maint_select_Event              -- Subprogram SPEC & BODY
--  .
--  .....................................................................

procedure maint_select_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: maint_select.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: maint_select.

begin -- maint_select_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel maintain, parm maint_select: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
```

```
        Text_IO.Put_Line (Value(1));
      else
        Text_IO.Put_Line ("none");
      end if;

      -- maint_select
      Destroy_Panel;
      Panel_select_1.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

      -- MERGE NOTE: Add code BELOW this line for parm: maint_select.
      -- MERGE NOTE: Add code ABOVE this line for parm: maint_select.

    end maint_select_Event;



    --
    -- end EVENT HANDLERS
    --
    -------------------------------------------------------------------------



    --  ....................................................................
    --  .
    --  .    Dispatch_Item                      -- Subprogram BODY
    --  .
    --  ....................................................................

    procedure Dispatch_Item
      ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

    --| NOTES: (none)

    begin -- Dispatch_Item

      if TAE.Tae_Misc.s_equal ("maint_backup", User_Context_Ptr.Parm_Name) then
        maint_backup_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("maint_cancel", User_Context_Ptr.Parm_Name)
  then
        maint_cancel_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("maint_create", User_Context_Ptr.Parm_Name)
  then
        maint_create_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("maint_delete", User_Context_Ptr.Parm_Name)
  then
        maint_delete_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("maint_help", User_Context_Ptr.Parm_Name) then
        maint_help_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("maint_select", User_Context_Ptr.Parm_Name)
  then
        maint_select_Event (User_Context_Ptr);
      end if;
  end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_maintain;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         pan_mbn_kps_s.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--   ************************************************************************
--   *
--   *    Panel_mbn_kps                        -- Package SPEC
--   *
--   ************************************************************************

with TAE;
with X_Windows;

package Panel_mbn_kps is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  mbn_kps
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  mbn_kps
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;    -- panel information


  --   ................................................................
  --   .
  --   .    Initialize_Panel                  -- Subprogram SPEC
  --   .
--  ......................................................................

procedure Initialize_Panel
  ( Collection_Read                          -- TAE Collection read from
      : in TAE.Tae_Co.Collection_Ptr );      -- resource file

--| PURPOSE:
--| This procedure initializes the Info.Target and Info.View for this panel
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
--| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
--|    Collection_Read
--|
--| NOTES: (none)




--  ......................................................................
--  .
--  .    Create_Panel                        -- Subprogram SPEC
--  .
--  ......................................................................

procedure Create_Panel
  ( Panel_State                              -- Flags sent to Wpt_NewPanel.
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                          -- Panel origin is offset from
      : in X_Windows.Window                  -- this X Window.  Null_Window
        := X_Windows.Null_Window );          -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--  ......................................................................
--  .
--  .    Connect_Panel                       -- Subprogram SPEC
--  .
--  ......................................................................

procedure Connect_Panel
```

```
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                          -- Panel origin is offset from
      : in X_Windows.Window                  -- this X Window.  Null_Window
        := X_Windows.Null_Window );          -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--  ....................................................................
--  .
--  .    Destroy_Panel                       -- Subprogram SPEC
--  .
--  ....................................................................

procedure Destroy_Panel;

--| PURPOSE:
--| This procedure erases a panel from the screen and de-allocates the
--| associated panel object (not the target and view).
--|
--| EXCEPTIONS:
--| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
--|
--| NOTES:
--| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
--| in any Wpt call until it is created again.




--  ....................................................................
--  .
--  .    Dispatch_Item                       -- Subprogram SPEC
--  .
```

```
  --  .....................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr                        -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_mbn_kps;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_mbn_kps_b.a
-- *** Generated:   Feb 23 13:21:26 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ***************************************************************************
--  *
--  *    Panel_mbn_kps                        -- Package BODY
--  *
--  ***************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.
with Panel_sdetail;
with Panel_grnd_eq;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_mbn_kps is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  mbn_kps
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   kps_keyin_from,  kps_keyin_to,    mbn_keyin,       mbn_kps_cancel,
--|   mbn_kps_help,    mbn_kps_ok
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--| 23-Feb-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

  --  ..................................................................
  --  .
  --  .    Initialize_Panel                 -- Subprogram BODY
  --  .
  --  ..................................................................

  procedure Initialize_Panel
    ( Collection_Read
        : in TAE.Tae_Co.Collection_Ptr ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

  begin -- Initialize_Panel

    Info := new TAE.Tae_Wpt.Event_Context;
    Info.Collection := Collection_Read;
    TAE.Tae_Co.Co_Find (Info.Collection, "mbn_kps_v", Info.View);
    TAE.Tae_Co.Co_Find (Info.Collection, "mbn_kps_t", Info.Target);

    -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

  exception
  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_mbn_kps.Initialize_Panel:  "
      & "Collection_Read not initialized.");
    raise;

  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_mbn_kps.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  ......................................................................
--  .
--  .    Create_Panel                          -- Subprogram BODY
--  .
--  ......................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id              => Global.Default_Display_Id,
        Data_Vm                 => Info.Target,
        View_Vm                 => Info.View,
        Relative_Window         => Relative_Window,
        User_Context            => Info,
        Flags                   => Panel_State,
        Panel_Id                => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (mbn_kps) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception
```

```
  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_mbn_kps.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_mbn_kps.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  ......................................................................
--  .
--  .    Connect_Panel                      -- Subprogram BODY
--  .
--  ......................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window       => Relative_Window,
        Panel_State           => Panel_State );
  else
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_STATE =>
    Text_IO.Put_Line ("Panel_mbn_kps.Connect_Panel:  "
      & "Invalid panel state.");
    raise;
```

```
end Connect_Panel;



--  ...............................................................
--  .
--  .    Destroy_Panel                   -- Subprogram BODY
--  .
--  ...............................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_mbn_kps.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;



--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--  ...............................................................
--  .
--  .    kps_keyin_from_Event            -- Subprogram SPEC & BODY
--  .
--  ...............................................................

procedure kps_keyin_from_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is
```

```
    --| PURPOSE:
    --| EVENT HANDLER.  Insert application specific information.
    --|
    --| NOTES: (none)

      Value : array (1..1) of TAE.Taefloat;
      Count : TAE.Taeint;

      -- MERGE NOTE: Add declarations BELOW this line for parm: kps_keyin_from.
      -- MERGE NOTE: Add declarations ABOVE this line for parm: kps_keyin_from.

    begin -- kps_keyin_from_Event

      TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
      Text_IO.Put ("Panel mbn_kps, parm kps_keyin_from: value = ");
      if Count > 0 then
        TAE.Tae_Vm.Vm_Extract_RVAL (Info.Parm_Ptr, 1, Value(1));
        Global.Taefloat_IO.Put (Value(1));  Text_IO.New_Line;
      else
        Text_IO.Put_Line ("none");
      end if;

      -- MERGE NOTE: Add code BELOW this line for parm: kps_keyin_from.
  ----------------------------------------------------------------------
  -- ADDED by RUEY-WEN (VINCENT) HONG  960223
  ----------------------------------------------------------------------
      Global.kps_from := float(Value(1));


      -- MERGE NOTE: Add code ABOVE this line for parm: kps_keyin_from.

    end kps_keyin_from_Event;



    --  ......................................................................
    --  .
    --  .    kps_keyin_to_Event                -- Subprogram SPEC & BODY
    --  .
    --  ......................................................................

    procedure kps_keyin_to_Event
      ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

    --| PURPOSE:
    --| EVENT HANDLER.  Insert application specific information.
    --|
    --| NOTES: (none)

      Value : array (1..1) of TAE.Taefloat;
      Count : TAE.Taeint;

      -- MERGE NOTE: Add declarations BELOW this line for parm: kps_keyin_to.
      -- MERGE NOTE: Add declarations ABOVE this line for parm: kps_keyin_to.
  begin -- kps_keyin_to_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel mbn_kps, parm kps_keyin_to: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_RVAL (Info.Parm_Ptr, 1, Value(1));
      Global.Taefloat_IO.Put (Value(1));  Text_IO.New_Line;
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: kps_keyin_to.
------------------------------------------------------------------------
-- ADDED by RUEY-WEN (VINCENT) HONG  960223
------------------------------------------------------------------------
    Global.kps_to := float(Value(1));

    -- MERGE NOTE: Add code ABOVE this line for parm: kps_keyin_to.

  end kps_keyin_to_Event;



  --  ..................................................................
  --  .
  --  .   mbn_keyin_Event                   -- Subprogram SPEC & BODY
  --  .
  --  ..................................................................

  procedure mbn_keyin_Event
    ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| PURPOSE:
  --| EVENT HANDLER.  Insert application specific information.
  --|
  --| NOTES: (none)

    Value : array (1..1) of TAE.Taeint;
    Count : TAE.Taeint;

    -- MERGE NOTE: Add declarations BELOW this line for parm: mbn_keyin.
    -- MERGE NOTE: Add declarations ABOVE this line for parm: mbn_keyin.

  begin -- mbn_keyin_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel mbn_kps, parm mbn_keyin: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_IVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (TAE.Taeint'image(Value(1)));
    else
      Text_IO.Put_Line ("none");
    end if;
    -- MERGE NOTE: Add code BELOW this line for parm: mbn_keyin.
----------------------------------------------------------------------
-- ADDED by RUEY-WEN (VINCENT) HONG  960223
----------------------------------------------------------------------
    Global.mbn := natural(Value(1));
    -- MERGE NOTE: Add code ABOVE this line for parm: mbn_keyin.

  end mbn_keyin_Event;



  --  ...................................................................
  --  .
  --  .    mbn_kps_cancel_Event             -- Subprogram SPEC & BODY
  --  .
  --  ...................................................................

  procedure mbn_kps_cancel_Event
    ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| PURPOSE:
  --| EVENT HANDLER.  Insert application specific information.
  --|
  --| NOTES: (none)

    Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
    Count : TAE.Taeint;

    -- MERGE NOTE: Add declarations BELOW this line for parm: mbn_kps_cancel.
    -- MERGE NOTE: Add declarations ABOVE this line for parm: mbn_kps_cancel.

  begin -- mbn_kps_cancel_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel mbn_kps, parm mbn_kps_cancel: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- mbn_kps_cancel
    Destroy_Panel;
    Panel_sdetail.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

    -- MERGE NOTE: Add code BELOW this line for parm: mbn_kps_cancel.
    -- MERGE NOTE: Add code ABOVE this line for parm: mbn_kps_cancel.

  end mbn_kps_cancel_Event;
```

```
--  ...........................................................
--  .
--  .    mbn_kps_help_Event              -- Subprogram SPEC & BODY
--  .
--  ...........................................................

procedure mbn_kps_help_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: mbn_kps_help.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: mbn_kps_help.

begin -- mbn_kps_help_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel mbn_kps, parm mbn_kps_help: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- mbn_kps_help
  Destroy_Panel;
  Panel_sdetail.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: mbn_kps_help.
  -- MERGE NOTE: Add code ABOVE this line for parm: mbn_kps_help.

end mbn_kps_help_Event;



--  ...........................................................
--  .
--  .    mbn_kps_ok_Event                -- Subprogram SPEC & BODY
--  .
--  ...........................................................

procedure mbn_kps_ok_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

    Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
    Count : TAE.Taeint;

    -- MERGE NOTE: Add declarations BELOW this line for parm: mbn_kps_ok.
    -- MERGE NOTE: Add declarations ABOVE this line for parm: mbn_kps_ok.

  begin -- mbn_kps_ok_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel mbn_kps, parm mbn_kps_ok: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- mbn_kps_ok
    Destroy_Panel;
    Panel_grnd_eq.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

    -- MERGE NOTE: Add code BELOW this line for parm: mbn_kps_ok.
    -- MERGE NOTE: Add code ABOVE this line for parm: mbn_kps_ok.

  end mbn_kps_ok_Event;



  --
  -- end EVENT HANDLERS
  --
  -------------------------------------------------------------------------



  --   ......................................................................
  --   .
  --   .    Dispatch_Item                    -- Subprogram BODY
  --   .
  --   ......................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| NOTES: (none)

  begin -- Dispatch_Item

    if TAE.Tae_Misc.s_equal ("kps_keyin_from", User_Context_Ptr.Parm_Name)
then
      kps_keyin_from_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("kps_keyin_to", User_Context_Ptr.Parm_Name)
```

```
then
      kps_keyin_to_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("mbn_keyin", User_Context_Ptr.Parm_Name) then
      mbn_keyin_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("mbn_kps_cancel",
User_Context_Ptr.Parm_Name) then
      mbn_kps_cancel_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("mbn_kps_help", User_Context_Ptr.Parm_Name)
then
      mbn_kps_help_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("mbn_kps_ok", User_Context_Ptr.Parm_Name) then
      mbn_kps_ok_Event (User_Context_Ptr);
    end if;

  end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_mbn_kps;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         pan_op_no_s.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--   ***************************************************************************
--   *
--   *    Panel_op_no                          -- Package SPEC
--   *
--   ***************************************************************************

with TAE;
with X_Windows;

package Panel_op_no is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  op_no
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
```

```
--|     The panel's name is changed (not title)
--| For panel:  op_no
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;   -- panel information


  --  ...................................................................
  --  .
  --  .    Initialize_Panel                 -- Subprogram SPEC
  --  .
  --  ...................................................................

  procedure Initialize_Panel
    ( Collection_Read                       -- TAE Collection read from
        : in TAE.Tae_Co.Collection_Ptr );   -- resource file

  --| PURPOSE:
  --| This procedure initializes the Info.Target and Info.View for this panel
  --|
  --| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
  --| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
  --|    Collection_Read
  --|
  --| NOTES: (none)




  --  ...................................................................
  --  .
  --  .    Create_Panel                     -- Subprogram SPEC
  --  .
  --  ...................................................................

  procedure Create_Panel
    ( Panel_State                           -- Flags sent to Wpt_NewPanel.
        : in TAE.Tae_Wpt.Wpt_Flags
          := TAE.Tae_Wpt.WPT_PREFERRED;

      Relative_Window                       -- Panel origin is offset from
        : in X_Windows.Window               -- this X Window.  Null_Window
          := X_Windows.Null_Window );       -- uses the root window.

  --| PURPOSE:
  --| This procedure creates this panel object in the specified Panel_State
  --| and stores the panel Id in Info.Panel_Id.
  --|
  --| EXCEPTIONS:
```

```
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--  ..................................................................
--  .
--  .    Connect_Panel                      -- Subprogram SPEC
--  .
--  ..................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                         -- Panel origin is offset from
      : in X_Windows.Window                 -- this X Window.  Null_Window
        := X_Windows.Null_Window );         -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--  ..................................................................
--  .
--  .    Destroy_Panel                      -- Subprogram SPEC
--  .
--  ..................................................................

procedure Destroy_Panel;

--| PURPOSE:
--| This procedure erases a panel from the screen and de-allocates the
--| associated panel object (not the target and view).
--|
--| EXCEPTIONS:
  --| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
  --|
  --| NOTES:
  --| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
  --| in any Wpt call until it is created again.




  --   ..................................................................
  --   .
  --   .    Dispatch_Item                      -- Subprogram SPEC
  --   .
  --   ..................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr                       -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_op_no;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_op_no_b.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--   ************************************************************************
--   *
--   *    Panel_op_no                        -- Package BODY
--   *
--   ************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.
with Panel_keyword;
with Panel_opdetail;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_op_no is
```

```
--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  op_no
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   keyin_op_no,      op_no_cancel,     op_no_help,       op_no_label,
--|   op_no_ok,         test_case_label,  testcase_no
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

  --  ............................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram BODY
  --  .
  --  ............................................................

  procedure Initialize_Panel
    ( Collection_Read
        : in TAE.Tae_Co.Collection_Ptr ) is

  --| NOTES: (none)
  -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

begin -- Initialize_Panel

  Info := new TAE.Tae_Wpt.Event_Context;
  Info.Collection := Collection_Read;
  TAE.Tae_Co.Co_Find (Info.Collection, "op_no_v", Info.View);
  TAE.Tae_Co.Co_Find (Info.Collection, "op_no_t", Info.Target);

  -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_op_no.Initialize_Panel:  "
      & "Collection_Read not initialized.");
    raise;

  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_op_no.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  .....................................................................
--  .
--  .    Create_Panel                      -- Subprogram BODY
--  .
--  .....................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id           => Global.Default_Display_Id,
        Data_Vm                  => Info.Target,
        View_Vm                  => Info.View,
        Relative_Window          => Relative_Window,
        User_Context             => Info,
        Flags                    => Panel_State,
        Panel_Id                 => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (op_no) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_op_no.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_op_no.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  ..................................................................
--  .
--  .    Connect_Panel                    -- Subprogram BODY
--  .
--  ..................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window       => Relative_Window,
        Panel_State           => Panel_State );
  else
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_STATE =>
    Text_IO.Put_Line ("Panel_op_no.Connect_Panel:  "
      & "Invalid panel state.");
    raise;

end Connect_Panel;



--  .....................................................................
--  .
--  .    Destroy_Panel                   -- Subprogram BODY
--  .
--  .....................................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_op_no.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;
```

```
--++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--  ......................................................................
--  .
--  .    keyin_op_no_Event                -- Subprogram SPEC & BODY
--  .
--  ......................................................................

procedure keyin_op_no_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of TAE.Taeint;
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: keyin_op_no.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: keyin_op_no.

begin -- keyin_op_no_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel op_no, parm keyin_op_no: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_IVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (TAE.Taeint'image(Value(1)));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: keyin_op_no.
  -- MERGE NOTE: Add code ABOVE this line for parm: keyin_op_no.

end keyin_op_no_Event;



--  ......................................................................
--  .
--  .    op_no_cancel_Event               -- Subprogram SPEC & BODY
--  .
--  ......................................................................

procedure op_no_cancel_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
```

```
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: op_no_cancel.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: op_no_cancel.

begin -- op_no_cancel_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel op_no, parm op_no_cancel: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- op_no_cancel
  Destroy_Panel;
  Panel_keyword.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: op_no_cancel.
  -- MERGE NOTE: Add code ABOVE this line for parm: op_no_cancel.

end op_no_cancel_Event;


--  ...................................................................
--  .
--  .    op_no_help_Event                  -- Subprogram SPEC & BODY
--  .
--  ...................................................................

procedure op_no_help_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: op_no_help.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: op_no_help.

begin -- op_no_help_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel op_no, parm op_no_help: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- op_no_help
  Destroy_Panel;
  Panel_keyword.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: op_no_help.
  -- MERGE NOTE: Add code ABOVE this line for parm: op_no_help.

end op_no_help_Event;



--  ....................................................................
--  .
--  .    op_no_label_Event                -- Subprogram SPEC & BODY
--  .
--  ....................................................................

procedure op_no_label_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: op_no_label.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: op_no_label.

begin -- op_no_label_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel op_no, parm op_no_label: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: op_no_label.
  -- MERGE NOTE: Add code ABOVE this line for parm: op_no_label.

end op_no_label_Event;
```

```
--   ..............................................................
--   .
--   .    op_no_ok_Event                  -- Subprogram SPEC & BODY
--   .
--   ..............................................................

procedure op_no_ok_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: op_no_ok.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: op_no_ok.

begin -- op_no_ok_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel op_no, parm op_no_ok: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- op_no_ok
  Destroy_Panel;
  Panel_opdetail.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: op_no_ok.
  -- MERGE NOTE: Add code ABOVE this line for parm: op_no_ok.

end op_no_ok_Event;



--   ..............................................................
--   .
--   .    test_case_label_Event           -- Subprogram SPEC & BODY
--   .
--   ..............................................................

procedure test_case_label_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is
```

```
--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: test_case_label.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: test_case_label.

begin -- test_case_label_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel op_no, parm test_case_label: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: test_case_label.
  -- MERGE NOTE: Add code ABOVE this line for parm: test_case_label.

end test_case_label_Event;



--  ....................................................................
--  .
--  .    testcase_no_Event                -- Subprogram SPEC & BODY
--  .
--  ....................................................................

procedure testcase_no_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of TAE.Taeint;
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: testcase_no.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: testcase_no.

begin -- testcase_no_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel op_no, parm testcase_no: value = ");
  if Count > 0 then
```

```
        TAE.Tae_Vm.Vm_Extract_IVAL (Info.Parm_Ptr, 1, Value(1));
        Text_IO.Put_Line (TAE.Taeint'image(Value(1)));
      else
        Text_IO.Put_Line ("none");
      end if;

      -- MERGE NOTE: Add code BELOW this line for parm: testcase_no.
      -- MERGE NOTE: Add code ABOVE this line for parm: testcase_no.

    end testcase_no_Event;



    --
    -- end EVENT HANDLERS
    --
    ---------------------------------------------------------------------------




    --  ......................................................................
    --  .
    --  .    Dispatch_Item                      -- Subprogram BODY
    --  .
    --  ......................................................................

    procedure Dispatch_Item
      ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

    --| NOTES: (none)

    begin -- Dispatch_Item

      if TAE.Tae_Misc.s_equal ("keyin_op_no", User_Context_Ptr.Parm_Name) then
        keyin_op_no_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("op_no_cancel", User_Context_Ptr.Parm_Name)
  then
        op_no_cancel_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("op_no_help", User_Context_Ptr.Parm_Name) then
        op_no_help_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("op_no_label", User_Context_Ptr.Parm_Name)
  then
        op_no_label_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("op_no_ok", User_Context_Ptr.Parm_Name) then
        op_no_ok_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("test_case_label", User_Context_Ptr.Parm_Name)
  then
        test_case_label_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("testcase_no", User_Context_Ptr.Parm_Name)
  then
        testcase_no_Event (User_Context_Ptr);
      end if;

    end Dispatch_Item;
```

```
-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_op_no;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         pan_opdetail_s.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ***************************************************************************
--  *
--  *    Panel_opdetail                        -- Package SPEC
--  *
--  ***************************************************************************

with TAE;
with X_Windows;

package Panel_opdetail is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  opdetail
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  opdetail
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated

  Info : TAE.Tae_Wpt.Event_Context_Ptr;    -- panel information


  --  .......................................................................
  --  .
  --  .    Initialize_Panel                   -- Subprogram SPEC
  --  .
  --  .......................................................................
```

```
procedure Initialize_Panel
  ( Collection_Read                           -- TAE Collection read from
      : in TAE.Tae_Co.Collection_Ptr );        -- resource file

--| PURPOSE:
--| This procedure initializes the Info.Target and Info.View for this panel
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
--| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
--|    Collection_Read
--|
--| NOTES: (none)




--   .....................................................................
--   .
--   .    Create_Panel                         -- Subprogram SPEC
--   .
--   .....................................................................

procedure Create_Panel
  ( Panel_State                               -- Flags sent to Wpt_NewPanel.
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                           -- Panel origin is offset from
      : in X_Windows.Window                   -- this X Window.  Null_Window
        := X_Windows.Null_Window );           -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--   .....................................................................
--   .
--   .    Connect_Panel                        -- Subprogram SPEC
--   .
--   .....................................................................

procedure Connect_Panel
  ( Panel_State
        : in TAE.Tae_Wpt.Wpt_Flags
          := TAE.Tae_Wpt.WPT_PREFERRED;

      Relative_Window                     -- Panel origin is offset from
        : in X_Windows.Window             -- this X Window.  Null_Window
          := X_Windows.Null_Window );     -- uses the root window.

  --| PURPOSE:
  --| If this panel doesn't exist, this procedure creates this panel object
  --|   in the specifiec Panel_State and stores the panel Id in
  --|   Info.Panel_Id.
  --| If this panel does exist, it is set to the specified Panel_State.
  --|   In this case, Relative_Window is ignored.
  --|
  --| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
  --|    not initialized
  --| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
  --|    created
  --| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
  --|    Panel_State is an invalid state
  --|
  --| NOTES: (none)



  --   ...........................................................
  --   .
  --   .    Destroy_Panel                 -- Subprogram SPEC
  --   .
  --   ...........................................................

  procedure Destroy_Panel;

  --| PURPOSE:
  --| This procedure erases a panel from the screen and de-allocates the
  --| associated panel object (not the target and view).
  --|
  --| EXCEPTIONS:
  --| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
  --|
  --| NOTES:
  --| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
  --| in any Wpt call until it is created again.



  --   ...........................................................
  --   .
  --   .    Dispatch_Item                 -- Subprogram SPEC
  --   .
  --   ...........................................................
  procedure Dispatch_Item
    ( User_Context_Ptr                      -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_opdetail;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_opdetail_b.a
-- *** Generated:   Feb 28 15:30:54 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--   ************************************************************************
--   *
--   *    Panel_opdetail                    -- Package BODY
--   *
--   ************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

use Text_io, Global;


-- One "with" statement for each connected panel.
with Panel_op_no;
with Panel_sdetail;
with Panel_opdetail;
with Panel_mbn_kps;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_opdetail is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  opdetail
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   keyin_op_name,   opdetail_cancel, opdetail_getsig, opdetail_help,
--|   opdetail_next,   opdetail_ok,     profile_list
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--| 28-Feb-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

 -----------------------------------------------------------------------
 --ADDED by Ruey_Wen (Vincent) Hong 951128
 -----------------------------------------------------------------------

  Type operator_type is record
       op_name : string(1..TAE.Tae_Taeconf.STRINGSIZE);
       profile_pattern : String(1..TAE.Tae_Taeconf.STRINGSIZE);
  end record;

  type op_array is array(1..30) of operator_type;

  Operator : op_array;

  profile_vec       : s_vector(1..100):= (others=> new STRING(1..80));
  profileadd_vec    : s_vector(1..15)  := (others=> new STRING(1..80));
  profile_selected  : String (1..80)   := (others=>' ');
  profile_list      : file_type;
  profile_file_name : String(1..12) := "profile_list";

  number,
  index     : integer := 1;
  op_index  : integer := 1;
```

```
-----------------------------------------------------------------------

--  .....................................................................
--  .
--  .    Initialize_Panel                    -- Subprogram BODY
--  .
--  .....................................................................

procedure Initialize_Panel
  ( Collection_Read
      : in TAE.Tae_Co.Collection_Ptr ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

begin -- Initialize_Panel

  Info := new TAE.Tae_Wpt.Event_Context;
  Info.Collection := Collection_Read;
  TAE.Tae_Co.Co_Find (Info.Collection, "opdetail_v", Info.View);
  TAE.Tae_Co.Co_Find (Info.Collection, "opdetail_t", Info.Target);

  -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_opdetail.Initialize_Panel:  "
      & "Collection_Read not initialized.");
    raise;

  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_opdetail.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  .....................................................................
--  .
--  .    Create_Panel                        -- Subprogram BODY
--  .
--  .....................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;
    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id            => Global.Default_Display_Id,
        Data_Vm               => Info.Target,
        View_Vm               => Info.View,
        Relative_Window       => Relative_Window,
        User_Context          => Info,
        Flags                 => Panel_State,
        Panel_Id              => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (opdetail) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_opdetail.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_opdetail.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  ..................................................................
--  .
--  .    Connect_Panel                       -- Subprogram BODY
--  .
--  ..................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;
      Relative_Window
        : in X_Windows.Window
          := X_Windows.Null_Window ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

------------------------------------------------------------
 --ADDED By RUEY-WEN HONG (VINCENT) 960228
------------------------------------------------------------
   Dummy:Boolean;
   L    :integer:=1;
------------------------------------------------------------

  begin -- Connect_Panel

    if Info.Panel_Id = Tae.Null_Panel_Id then
      Create_Panel
        ( Relative_Window       => Relative_Window,
          Panel_State           => Panel_State );
    else
      TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
    end if;

    -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
    -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

------------------------------------------------------------------------
  -- ADDED By RUEY-WEN HONG (VINCENT) 960228
------------------------------------------------------------------------
  strlen(library,L);

  list_components(profile_list,profile_file_name,profile_vec,number);
  TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);

TAE.Tae_Wpt.Wpt_SetStringConstraints(Info.Panel_Id,"profile_list",taeint(numb
er),profile_vec);

  Dummy:=TAE.Tae_Wpt.Wpt_Pending;
------------------------------------------------------------------------


  exception

    when TAE.Tae_Wpt.BAD_STATE =>
      Text_IO.Put_Line ("Panel_opdetail.Connect_Panel:  "
        & "Invalid panel state.");
      raise;

  end Connect_Panel;
```

```
-- .................................................................
-- .
-- .    Destroy_Panel                    -- Subprogram BODY
-- .
-- .................................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_opdetail.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;



--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
-- .................................................................
-- .
-- .    keyin_op_name_Event              -- Subprogram SPEC & BODY
-- .
-- .................................................................

procedure keyin_op_name_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: keyin_op_name.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: keyin_op_name.

begin -- keyin_op_name_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel opdetail, parm keyin_op_name: value = ");
  if Count > 0 then

   -------------------------------------------------------
   -- ADDED By RUEY-WEN HONG (VINCENT) 960228
   -------------------------------------------------------
    Value(1) := (others => ' ');
   -------------------------------------------------------

    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: keyin_op_name.
  -- MERGE NOTE: Add code ABOVE this line for parm: keyin_op_name.

-------------------------------------------------------
-- ADDED By RUEY-WEN HONG (VINCENT) 960228
-------------------------------------------------------
Operator(op_index).op_name := Value(1);
-------------------------------------------------------

end keyin_op_name_Event;



--  .........................................................................
--  .
--  .    opdetail_cancel_Event            -- Subprogram SPEC & BODY
--  .
--  .........................................................................

procedure opdetail_cancel_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
```

```
   Count  : TAE.Taeint;

   -- MERGE NOTE: Add declarations BELOW this line for parm: opdetail_cancel.
   -- MERGE NOTE: Add declarations ABOVE this line for parm: opdetail_cancel.

begin -- opdetail_cancel_Event

   TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
   Text_IO.Put ("Panel opdetail, parm opdetail_cancel: value = ");
   if Count > 0 then
     TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
     Text_IO.Put_Line (Value(1));
   else
     Text_IO.Put_Line ("none");
   end if;

   -- opdetail_cancel
   Destroy_Panel;
   Panel_op_no.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

   -- MERGE NOTE: Add code BELOW this line for parm: opdetail_cancel.
   -- MERGE NOTE: Add code ABOVE this line for parm: opdetail_cancel.

end opdetail_cancel_Event;



--  ......................................................................
--  .
--  .    opdetail_getsig_Event            -- Subprogram SPEC & BODY
--  .
--  ......................................................................

procedure opdetail_getsig_Event
   ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

   Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
   Count : TAE.Taeint;

   -- MERGE NOTE: Add declarations BELOW this line for parm: opdetail_getsig.
   -- MERGE NOTE: Add declarations ABOVE this line for parm: opdetail_getsig.

begin -- opdetail_getsig_Event

   TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
   Text_IO.Put ("Panel opdetail, parm opdetail_getsig: value = ");
   if Count > 0 then

    -------------------------------------------------------
```

```
 -- ADDED By RUEY-WEN HONG (VINCENT) 960228
 ---------------------------------------------------------
  Value(1) := (others => ' ');
 ---------------------------------------------------------


  TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
  Text_IO.Put_Line (Value(1));
else
  Text_IO.Put_Line ("none");
end if;
----------------------------------------------------------------------
-- ADDED By RUEY-WEN HONG (VINCENT) 960228
----------------------------------------------------------------------
for J in 1..op_index loop
   text_io.new_line;
   text_io.put(Operator(J).op_name);
   text_io.new_line;
   text_io.put(Operator(J).profile_pattern);
end loop;

op_index := op_index + 1;



----------------------------------------------------------------------
-- ADDED By RUEY-WEN HONG (VINCENT) 960228
----------------------------------------------------------------------
for J in 1..op_index loop
   for K in 1..TAE.Tae_Taeconf.STRINGSIZE loop

       case Operator(J).profile_pattern(K) is

          when 'A' =>
                        If Global.Sig_Exist_Chk_Vec(1) = "0"
                        then
                          Global.Sig_Keep_Vec(Global.skv_index) := "A";
                          Global.skv_index := Global.skv_index + 1;
                          Global.Sig_Exist_Chk_Vec(1) := "1";
                        else
                          Null;
                        end if;

          when 'B' =>
                        If
                          Sig_Exist_Chk_Vec(2) = "0"
                        then
                          Sig_Keep_Vec(skv_index) := "B";
                          skv_index := skv_index + 1;
                          Sig_Exist_Chk_Vec(2) := "1";
                        else
                          Null;
                        end if;
          when 'C' =>
```

```
                        If
                          Sig_Exist_Chk_Vec(3) = "0"
                        then
                          Sig_Keep_Vec(skv_index) := "C";
                          skv_index := skv_index + 1;
                          Sig_Exist_Chk_Vec(3) := "1";
                        else
                          Null;
                        end if;
             when 'D' =>
                        If
                          Sig_Exist_Chk_Vec(4) = "0"
                        then
                          Sig_Keep_Vec(skv_index) := "D";
                          skv_index := skv_index + 1;
                          Sig_Exist_Chk_Vec(4) := "1";
                        else
                          Null;
                        end if;
            when 'E' =>
                       If
                          Sig_Exist_Chk_Vec(5) = "0"
                        then
                          Sig_Keep_Vec(skv_index) := "E";
                          skv_index := skv_index + 1;
                          Sig_Exist_Chk_Vec(5) := "1";
                        else
                          Null;
                        end if;
              when 'F' =>
                        If
                          Sig_Exist_Chk_Vec(6) = "0"
                        then
                          Sig_Keep_Vec(skv_index) := "F";
                          skv_index := skv_index + 1;
                          Sig_Exist_Chk_Vec(6) := "1";
                        else
                          Null;
                        end if;

              when others => Null;

           end case;
      end loop;
   end loop;


   -- opdetail_getsig
   Panel_sdetail.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

   -- MERGE NOTE: Add code BELOW this line for parm: opdetail_getsig.
   -- MERGE NOTE: Add code ABOVE this line for parm: opdetail_getsig.

 end opdetail_getsig_Event;
```

```
--  ...................................................................
--  .
--  .    opdetail_help_Event              -- Subprogram SPEC & BODY
--  .
--  ...................................................................

procedure opdetail_help_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: opdetail_help.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: opdetail_help.

begin -- opdetail_help_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel opdetail, parm opdetail_help: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- opdetail_help
  Destroy_Panel;
  Panel_op_no.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: opdetail_help.
  -- MERGE NOTE: Add code ABOVE this line for parm: opdetail_help.

end opdetail_help_Event;



--  ...................................................................
--  .
--  .    opdetail_next_Event              -- Subprogram SPEC & BODY
--  .
--  ...................................................................

procedure opdetail_next_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is
```

```
--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: opdetail_next.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: opdetail_next.

begin -- opdetail_next_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel opdetail, parm opdetail_next: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- opdetail_next
  Destroy_Panel;
  Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: opdetail_next.
  -- MERGE NOTE: Add code ABOVE this line for parm: opdetail_next.

end opdetail_next_Event;



--  ......................................................................
--  .
--  .    opdetail_ok_Event                -- Subprogram SPEC & BODY
--  .
--  ......................................................................

procedure opdetail_ok_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: opdetail_ok.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: opdetail_ok.

begin -- opdetail_ok_Event
```

```
   TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
   Text_IO.Put ("Panel opdetail, parm opdetail_ok: value = ");
   if Count > 0 then
     TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
     Text_IO.Put_Line (Value(1));
   else
     Text_IO.Put_Line ("none");
   end if;

   -- opdetail_ok
   Destroy_Panel;
   Panel_mbn_kps.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

   -- MERGE NOTE: Add code BELOW this line for parm: opdetail_ok.
   -- MERGE NOTE: Add code ABOVE this line for parm: opdetail_ok.

 end opdetail_ok_Event;



 --  ..................................................................
 --  .
 --  .    profile_list_Event              -- Subprogram SPEC & BODY
 --  .
 --  ..................................................................

 procedure profile_list_Event
   ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

 --| PURPOSE:
 --| EVENT HANDLER.  Insert application specific information.
 --|
 --| NOTES: (none)

   Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
   Count : TAE.Taeint;

   -- MERGE NOTE: Add declarations BELOW this line for parm: profile_list.
   -- MERGE NOTE: Add declarations ABOVE this line for parm: profile_list.

---------------------------------------------------------------
--ADDED By RUEY-WEN HONG (VINCENT) 960228
---------------------------------------------------------------
   S_Value : TAE.s_vector(1..1);
---------------------------------------------------------------


 begin -- profile_list_Event

   TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
   Text_IO.Put ("Panel opdetail, parm profile_list: value = ");
   if Count > 0 then
     TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
```

```
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: profile_list.
    -- MERGE NOTE: Add code ABOVE this line for parm: profile_list.
 ---------------------------------------------------------------
 --ADDED By RUEY-WEN HONG (VINCENT) 960228
 ---------------------------------------------------------------

--     profile_selected(1..80):=Value(1)(1..80);
--     profileadd_vec(index).all:=profile_selected;
--     TAE.Tae_Wpt.Wpt_PanelMessage(map_pro_display,"Test");
--    TAE.Tae_Wpt.Wpt_ParmReject(Info.Panel_Id,"map_pro_display","Test");
--    TAE.Tae_Wpt.Wpt_SetString(Info.Panel_Id,"map_pro_display","Test");
--     TAE.Tae_Wpt.Wpt_SetStringConstraints(
--        Info.Panel_Id,"selected_sop",taeint(index),profileadd_vec);
--     index := index + 1;

  S_Value(1) := new String (1..TAE.Tae_Taeconf.STRINGSIZE);

  S_Value(1).all := Value(1);

  TAE.TAE_Vm.Vm_SetString_Vec(Info.target, "map_pro_display", 1, S_Value,
TAE.TAE_Vm.P_Update);
  TAE.TAE_Wpt.Wpt_ParmUpdate(Info.Panel_Id, "map_pro_display");

  -------------------------------------------------------
  -- ADDED
  -------------------------------------------------------
  Operator(op_index).profile_pattern := Value(1);
  -------------------------------------------------------

  end profile_list_Event;



  --
  -- end EVENT HANDLERS
  --
  --------------------------------------------------------------------------



  --  ......................................................................
  --  .
  --  .    Dispatch_Item                     -- Subprogram BODY
  --  .
  --  ......................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is
  --| NOTES: (none)

  begin -- Dispatch_Item

     if TAE.Tae_Misc.s_equal ("keyin_op_name", User_Context_Ptr.Parm_Name) then
       keyin_op_name_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("opdetail_cancel", User_Context_Ptr.Parm_Name)
then
       opdetail_cancel_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("opdetail_getsig", User_Context_Ptr.Parm_Name)
then
       opdetail_getsig_Event (User_Context_Ptr);
     elsif TAE.Tae_Misc.s_equal ("opdetail_help", User_Context_Ptr.Parm_Name)
then
       opdetail_help_Event (User_Context_Ptr);
     elsif TAE.Tae_Misc.s_equal ("opdetail_next", User_Context_Ptr.Parm_Name)
then
       opdetail_next_Event (User_Context_Ptr);
     elsif TAE.Tae_Misc.s_equal ("opdetail_ok", User_Context_Ptr.Parm_Name)
then
       opdetail_ok_Event (User_Context_Ptr);
     elsif TAE.Tae_Misc.s_equal ("profile_list", User_Context_Ptr.Parm_Name)
then
       profile_list_Event (User_Context_Ptr);
     end if;

  end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_opdetail;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         pan_query_s.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--   *************************************************************************
--   *
--   *    Panel_query                        -- Package SPEC
--   *
--   *************************************************************************

with TAE;
with X_Windows;

package Panel_query is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  query
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  query
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;   -- panel information


  --  ..................................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram SPEC
  --  .
  --  ..................................................................

  procedure Initialize_Panel
    ( Collection_Read                        -- TAE Collection read from
        : in TAE.Tae_Co.Collection_Ptr );    -- resource file

  --| PURPOSE:
  --| This procedure initializes the Info.Target and Info.View for this panel
  --|
  --| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
  --| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
  --|    Collection_Read
  --|
  --| NOTES: (none)




  --  ..................................................................
  --  .
  --  .    Create_Panel                      -- Subprogram SPEC
  --  .
  --  ..................................................................

  procedure Create_Panel
    ( Panel_State                            -- Flags sent to Wpt_NewPanel.
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                        -- Panel origin is offset from
      : in X_Windows.Window                -- this X Window.  Null_Window
        := X_Windows.Null_Window );        -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--  .......................................................................
--  .
--  .    Connect_Panel                     -- Subprogram SPEC
--  .
--  .......................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                        -- Panel origin is offset from
      : in X_Windows.Window                -- this X Window.  Null_Window
        := X_Windows.Null_Window );        -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--  .......................................................................
```

```
  --  .
  --  .    Destroy_Panel                        -- Subprogram SPEC
  --  .
  --  ...................................................................

  procedure Destroy_Panel;

  --| PURPOSE:
  --| This procedure erases a panel from the screen and de-allocates the
  --| associated panel object (not the target and view).
  --|
  --| EXCEPTIONS:
  --| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
  --|
  --| NOTES:
  --| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
  --| in any Wpt call until it is created again.




  --  ...................................................................
  --  .
  --  .    Dispatch_Item                        -- Subprogram SPEC
  --  .
  --  ...................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr                          -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_query;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_query_b.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  **************************************************************************
--  *
--  *    Panel_query                          -- Package BODY
--  *
--  **************************************************************************
```

```
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.
with Panel_sb_main;
with Panel_keyword;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_query is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  query
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   profile_label,   query_cancel,    query_help,      query_import,
--|   query_manual
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.
-- ...........................................................
-- .
-- .    Initialize_Panel                  -- Subprogram BODY
-- .
-- ...........................................................

procedure Initialize_Panel
  ( Collection_Read
      : in TAE.Tae_Co.Collection_Ptr ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

begin -- Initialize_Panel

  Info := new TAE.Tae_Wpt.Event_Context;
  Info.Collection := Collection_Read;
  TAE.Tae_Co.Co_Find (Info.Collection, "query_v", Info.View);
  TAE.Tae_Co.Co_Find (Info.Collection, "query_t", Info.Target);

  -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
  -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_query.Initialize_Panel:  "
      & "Collection_Read not initialized.");
    raise;

  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_query.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



-- ...........................................................
-- .
-- .    Create_Panel                      -- Subprogram BODY
-- .
-- ...........................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
```

```
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id            => Global.Default_Display_Id,
        Data_Vm               => Info.Target,
        View_Vm               => Info.View,
        Relative_Window       => Relative_Window,
        User_Context          => Info,
        Flags                 => Panel_State,
        Panel_Id              => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (query) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_query.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_query.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  ......................................................................
--  .
--  .    Connect_Panel                       -- Subprogram BODY
--  .
--  ......................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
```

```
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window       => Relative_Window,
        Panel_State           => Panel_State );
  else
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_STATE =>
    Text_IO.Put_Line ("Panel_query.Connect_Panel:  "
      & "Invalid panel state.");
    raise;

end Connect_Panel;



--  .....................................................................
--  .
--  .    Destroy_Panel                    -- Subprogram BODY
--  .
--  .....................................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
```

```
    Text_IO.Put_Line ("Panel_query.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;



--++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--  .....................................................................
--  .
--  .    profile_label_Event              -- Subprogram SPEC & BODY
--  .
--  .....................................................................

procedure profile_label_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: profile_label.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: profile_label.

begin -- profile_label_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel query, parm profile_label: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: profile_label.
  -- MERGE NOTE: Add code ABOVE this line for parm: profile_label.

end profile_label_Event;
```

```
--  ...........................................................
--  .
--  .    query_cancel_Event              -- Subprogram SPEC & BODY
--  .
--  ...........................................................

procedure query_cancel_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: query_cancel.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: query_cancel.

begin -- query_cancel_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel query, parm query_cancel: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- query_cancel
  Destroy_Panel;
  Panel_sb_main.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: query_cancel.
  -- MERGE NOTE: Add code ABOVE this line for parm: query_cancel.

end query_cancel_Event;



--  ...........................................................
--  .
--  .    query_help_Event                -- Subprogram SPEC & BODY
--  .
--  ...........................................................

procedure query_help_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
```

```
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: query_help.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: query_help.

begin -- query_help_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel query, parm query_help: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- query_help
  Destroy_Panel;
  Panel_sb_main.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: query_help.
  -- MERGE NOTE: Add code ABOVE this line for parm: query_help.

end query_help_Event;



--  ........................................................................
--  .
--  .    query_import_Event               -- Subprogram SPEC & BODY
--  .
--  ........................................................................

procedure query_import_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: query_import.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: query_import.

begin -- query_import_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
```

```
   Text_IO.Put ("Panel query, parm query_import: value = ");
   if Count > 0 then
     TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
     Text_IO.Put_Line (Value(1));
   else
     Text_IO.Put_Line ("none");
   end if;

   -- MERGE NOTE: Add code BELOW this line for parm: query_import.
   -- MERGE NOTE: Add code ABOVE this line for parm: query_import.

 end query_import_Event;



 --  .......................................................................
 --  .
 --  .    query_manual_Event                -- Subprogram SPEC & BODY
 --  .
 --  .......................................................................

 procedure query_manual_Event
   ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

 --| PURPOSE:
 --| EVENT HANDLER.  Insert application specific information.
 --|
 --| NOTES: (none)

   Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
   Count : TAE.Taeint;

   -- MERGE NOTE: Add declarations BELOW this line for parm: query_manual.
   -- MERGE NOTE: Add declarations ABOVE this line for parm: query_manual.

 begin -- query_manual_Event

   TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
   Text_IO.Put ("Panel query, parm query_manual: value = ");
   if Count > 0 then
     TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
     Text_IO.Put_Line (Value(1));
   else
     Text_IO.Put_Line ("none");
   end if;

   -- query_manual
   Connect_Panel (TAE.Tae_Wpt.WPT_INVISIBLE);
   Panel_keyword.Connect_Panel (TAE.Tae_Wpt.WPT_VISIBLE);

   -- MERGE NOTE: Add code BELOW this line for parm: query_manual.
   -- MERGE NOTE: Add code ABOVE this line for parm: query_manual.

 end query_manual_Event;
```

```
   --
   -- end EVENT HANDLERS
   --
   -------------------------------------------------------------------------



   --   ...................................................................
   --   .
   --   .    Dispatch_Item                      -- Subprogram BODY
   --   .
   --   ...................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| NOTES: (none)

  begin -- Dispatch_Item

    if TAE.Tae_Misc.s_equal ("profile_label", User_Context_Ptr.Parm_Name) then
      profile_label_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("query_cancel", User_Context_Ptr.Parm_Name)
then
      query_cancel_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("query_help", User_Context_Ptr.Parm_Name) then
      query_help_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("query_import", User_Context_Ptr.Parm_Name)
then
      query_import_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("query_manual", User_Context_Ptr.Parm_Name)
then
      query_manual_Event (User_Context_Ptr);
    end if;

  end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_query;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_sb_main_s.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--   *********************************************************************
--  *
--  *    Panel_sb_main                          -- Package SPEC
--  *
--  ************************************************************************

with TAE;
with X_Windows;

package Panel_sb_main is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  sb_main
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  sb_main
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;   -- panel information


  --  ...................................................................
  --  .
  --  .    Initialize_Panel                   -- Subprogram SPEC
  --  .
  --  ...................................................................

  procedure Initialize_Panel
    ( Collection_Read                         -- TAE Collection read from
        : in TAE.Tae_Co.Collection_Ptr );     -- resource file

  --| PURPOSE:
  --| This procedure initializes the Info.Target and Info.View for this panel
  --|
  --| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
  --| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
  --|    Collection_Read
  --|
  --| NOTES: (none)
```

```
--  .....................................................................
--  .
--  .    Create_Panel                      -- Subprogram SPEC
--  .
--  .....................................................................

procedure Create_Panel
  ( Panel_State                            -- Flags sent to Wpt_NewPanel.
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                        -- Panel origin is offset from
      : in X_Windows.Window                -- this X Window.  Null_Window
        := X_Windows.Null_Window );        -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--  .....................................................................
--  .
--  .    Connect_Panel                     -- Subprogram SPEC
--  .
--  .....................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                        -- Panel origin is offset from
      : in X_Windows.Window                -- this X Window.  Null_Window
        := X_Windows.Null_Window );        -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
  --| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
  --|    not initialized
  --| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
  --|    created
  --| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
  --|    Panel_State is an invalid state
  --|
  --| NOTES: (none)



  --  ...............................................................
  --  .
  --  .    Destroy_Panel                    -- Subprogram SPEC
  --  .
  --  ...............................................................

  procedure Destroy_Panel;

  --| PURPOSE:
  --| This procedure erases a panel from the screen and de-allocates the
  --| associated panel object (not the target and view).
  --|
  --| EXCEPTIONS:
  --| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
  --|
  --| NOTES:
  --| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
  --| in any Wpt call until it is created again.




  --  ...............................................................
  --  .
  --  .    Dispatch_Item                    -- Subprogram SPEC
  --  .
  --  ...............................................................

  procedure Dispatch_Item
    ( User_Context_Ptr                     -- Wpt Event Context for a PARM
       : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_sb_main;
```

```
-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_sb_main_b.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author     : Ruey-Wen (Vincent) Hong
-- *** Date       : Mar 5, 1996
-- *** Application: CAPS Software Base

--   **************************************************************************
--   *
--   *    Panel_sb_main                        -- Package BODY
--   *
--   **************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.
with Panel_maintain;
with Panel_query;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_sb_main is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  sb_main
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
```

```
--|   help_button,     main_label,      quit_button,     sb_maint_button,
--|   sb_query_button
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

  --  ......................................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram BODY
  --  .
  --  ......................................................................

  procedure Initialize_Panel
    ( Collection_Read
        : in TAE.Tae_Co.Collection_Ptr ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

  begin -- Initialize_Panel

    Info := new TAE.Tae_Wpt.Event_Context;
    Info.Collection := Collection_Read;
    TAE.Tae_Co.Co_Find (Info.Collection, "sb_main_v", Info.View);
    TAE.Tae_Co.Co_Find (Info.Collection, "sb_main_t", Info.Target);

    -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

  exception

    when TAE.UNINITIALIZED_PTR =>
      Text_IO.Put_Line ("Panel_sb_main.Initialize_Panel:  "
        & "Collection_Read not initialized.");
      raise;

    when TAE.Tae_Co.NO_SUCH_MEMBER =>
      Text_IO.Put_Line ("Panel_sb_main.Initialize_Panel:  "
        & "(View or Target) not in Collection.");
      raise;

  end Initialize_Panel;



  --  ......................................................................
--  .
--  .    Create_Panel                      -- Subprogram BODY
--  .
--  ......................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id            => Global.Default_Display_Id,
        Data_Vm               => Info.Target,
        View_Vm               => Info.View,
        Relative_Window       => Relative_Window,
        User_Context          => Info,
        Flags                 => Panel_State,
        Panel_Id              => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (sb_main) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_sb_main.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_sb_main.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;


--  ......................................................................
--  .
--  .    Connect_Panel                          -- Subprogram BODY
--  .
--  ......................................................................

procedure Connect_Panel
   ( Panel_State
       : in TAE.Tae_Wpt.Wpt_Flags
         := TAE.Tae_Wpt.WPT_PREFERRED;

     Relative_Window
       : in X_Windows.Window
         := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window       => Relative_Window,
        Panel_State           => Panel_State );
  else
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_STATE =>
    Text_IO.Put_Line ("Panel_sb_main.Connect_Panel:  "
      & "Invalid panel state.");
    raise;

end Connect_Panel;



--  ......................................................................
--  .
--  .    Destroy_Panel                          -- Subprogram BODY
--  .
--  ......................................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_sb_main.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;



--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--  ....................................................................
--  .
--  .    help_button_Event                -- Subprogram SPEC & BODY
--  .
--  ....................................................................

procedure help_button_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: help_button.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: help_button.

begin -- help_button_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel sb_main, parm help_button: value = ");
  if Count > 0 then
```

```
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: help_button.
  -- MERGE NOTE: Add code ABOVE this line for parm: help_button.

end help_button_Event;



--  .................................................................
--  .
--  .    main_label_Event                  -- Subprogram SPEC & BODY
--  .
--  .................................................................

procedure main_label_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: main_label.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: main_label.

begin -- main_label_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel sb_main, parm main_label: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: main_label.
  -- MERGE NOTE: Add code ABOVE this line for parm: main_label.

end main_label_Event;



--  .................................................................
--  .
--  .    quit_button_Event                 -- Subprogram SPEC & BODY
--  .
--  .............................................................

procedure quit_button_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: quit_button.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: quit_button.

begin -- quit_button_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel sb_main, parm quit_button: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- MERGE NOTE: Add code BELOW this line for parm: quit_button.
  -- MERGE NOTE: Add code ABOVE this line for parm: quit_button.

end quit_button_Event;



--  .............................................................
--  .
--  .    sb_maint_button_Event            -- Subprogram SPEC & BODY
--  .
--  .............................................................

procedure sb_maint_button_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: sb_maint_button.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: sb_maint_button.
```

```
begin -- sb_maint_button_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel sb_main, parm sb_maint_button: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- sb_maint_button
  Panel_maintain.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: sb_maint_button.
  -- MERGE NOTE: Add code ABOVE this line for parm: sb_maint_button.

end sb_maint_button_Event;



--  ...................................................................
--  .
--  .    sb_query_button_Event            -- Subprogram SPEC & BODY
--  .
--  ...................................................................

procedure sb_query_button_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: sb_query_button.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: sb_query_button.

begin -- sb_query_button_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel sb_main, parm sb_query_button: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- sb_query_button
```

```
      Panel_query.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

      -- MERGE NOTE: Add code BELOW this line for parm: sb_query_button.
      -- MERGE NOTE: Add code ABOVE this line for parm: sb_query_button.

    end sb_query_button_Event;



    --
    -- end EVENT HANDLERS
    --
    -------------------------------------------------------------------------



    --  .....................................................................
    --  .
    --  .    Dispatch_Item                     -- Subprogram BODY
    --  .
    --  .....................................................................

    procedure Dispatch_Item
      ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

    --| NOTES: (none)

    begin -- Dispatch_Item

      if TAE.Tae_Misc.s_equal ("help_button", User_Context_Ptr.Parm_Name) then
        help_button_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("main_label", User_Context_Ptr.Parm_Name) then
        main_label_Event (User_Context_Ptr);
      elsif TAE.Tae_Misc.s_equal ("quit_button", User_Context_Ptr.Parm_Name)
  then
        quit_button_Event (User_Context_Ptr);
     elsif TAE.Tae_Misc.s_equal ("sb_maint_button", User_Context_Ptr.Parm_Name)
  then
        sb_maint_button_Event (User_Context_Ptr);
     elsif TAE.Tae_Misc.s_equal ("sb_query_button", User_Context_Ptr.Parm_Name)
  then
        sb_query_button_Event (User_Context_Ptr);
      end if;

    end Dispatch_Item;


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

  end Panel_sb_main;

  -- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
  -- *** File:        pan_sdetail_s.a
```

```
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--  *************************************************************************
--  *
--  *    Panel_sdetail                      -- Package SPEC
--  *
--  *************************************************************************

with TAE;
with X_Windows;

package Panel_sdetail is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  sdetail
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  sdetail
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;    -- panel information


  --  ....................................................................
  --  .
  --  .    Initialize_Panel                   -- Subprogram SPEC
  --  .
  --  ....................................................................

  procedure Initialize_Panel
    ( Collection_Read                         -- TAE Collection read from
        : in TAE.Tae_Co.Collection_Ptr );     -- resource file

  --| PURPOSE:
  --| This procedure initializes the Info.Target and Info.View for this panel
  --|
  --| EXCEPTIONS:
```

```
--| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
--| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
--|    Collection_Read
--|
--| NOTES: (none)





--  ...................................................................
--  .
--  .    Create_Panel                   -- Subprogram SPEC
--  .
--  ...................................................................

procedure Create_Panel
  ( Panel_State                         -- Flags sent to Wpt_NewPanel.
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                     -- Panel origin is offset from
      : in X_Windows.Window             -- this X Window.  Null_Window
        := X_Windows.Null_Window );     -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--  ...................................................................
--  .
--  .    Connect_Panel                  -- Subprogram SPEC
--  .
--  ...................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                     -- Panel origin is offset from
      : in X_Windows.Window             -- this X Window.  Null_Window
        := X_Windows.Null_Window );     -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--  .......................................................................
--  .
--  .    Destroy_Panel                       -- Subprogram SPEC
--  .
--  .......................................................................

procedure Destroy_Panel;

--| PURPOSE:
--| This procedure erases a panel from the screen and de-allocates the
--| associated panel object (not the target and view).
--|
--| EXCEPTIONS:
--| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
--|
--| NOTES:
--| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
--| in any Wpt call until it is created again.





--  .......................................................................
--  .
--  .    Dispatch_Item                       -- Subprogram SPEC
--  .
--  .......................................................................

procedure Dispatch_Item
  ( User_Context_Ptr                     -- Wpt Event Context for a PARM
      : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

--| PURPOSE:
--| This procedure calls the Event Handler specified by User_Context_Ptr
--|
--| EXCEPTIONS:
    --| Application-specific
    --|
    --| NOTES: (none)

end Panel_sdetail;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_sdetail_b.a
-- *** Generated:   Feb 28 14:56:42 1996
-- *** Author     : Ruey-Wen (Vincent) Hong
-- *** Date       : Mar 5, 1996
-- *** Application: CAPS Software Base

--  ***************************************************************************
--  *
--  *    Panel_sdetail                        -- Package BODY
--  *
--  ***************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

use Text_io, Global;

-- One "with" statement for each connected panel.
with Panel_opdetail;
with Panel_sdetail;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_sdetail is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  sdetail
--|
--| The following WorkBench operations will also cause regeneration:
```

```
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   sdetail_cancel,  sdetail_help,    sdetail_next,    sdetail_ok,
--|   sig_display,     sig_list,        sig_name_keyin
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--| 28-Feb-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

  --  ...................................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram BODY
  --  .
  --  ...................................................................

  procedure Initialize_Panel
    ( Collection_Read
        : in TAE.Tae_Co.Collection_Ptr ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

  begin -- Initialize_Panel

    Info := new TAE.Tae_Wpt.Event_Context;
    Info.Collection := Collection_Read;
    TAE.Tae_Co.Co_Find (Info.Collection, "sdetail_v", Info.View);
    TAE.Tae_Co.Co_Find (Info.Collection, "sdetail_t", Info.Target);

    -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

  exception

    when TAE.UNINITIALIZED_PTR =>
      Text_IO.Put_Line ("Panel_sdetail.Initialize_Panel:  "
        & "Collection_Read not initialized.");
      raise;

    when TAE.Tae_Co.NO_SUCH_MEMBER =>
      Text_IO.Put_Line ("Panel_sdetail.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  ..................................................................
--  .
--  .    Create_Panel                          -- Subprogram BODY
--  .
--  ..................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id            => Global.Default_Display_Id,
        Data_Vm               => Info.Target,
        View_Vm               => Info.View,
        Relative_Window       => Relative_Window,
        User_Context          => Info,
        Flags                 => Panel_State,
        Panel_Id              => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (sdetail) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_sdetail.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_sdetail.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  .....................................................................
--  .
--  .    Connect_Panel                    -- Subprogram BODY
--  .
--  .....................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

---------------------------------------------------------
-- ADDED By RUEY-WEN (VINCENT) HONG 960228
---------------------------------------------------------
 Dummy:Boolean;
 S_Sig_Keep_vec : TAE.S_vector(1..10);
---------------------------------------------------------


begin -- Connect_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window       => Relative_Window,
        Panel_State           => Panel_State );
  else
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

------------------------------------------------------------------------
-- ADDED By RUEY-WEN (VINCENT) HONG 960228
------------------------------------------------------------------------

  TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
```

```
--   S_Sig_Keep_vec :=  (others=> new String(1..TAE.Tae_Taeconf.STRINGSIZE));

     S_Sig_Keep_vec :=  (others=> new String(1..1));

     for Count in 1 .. 10 loop
         S_Sig_Keep_vec(Count).all := Sig_Keep_vec(Count);
     end loop;

--    text_io.newline;
     if Sig_Keep_vec(1) = "A" then
        text_io.put("I got A!");
        text_io.new_line;
     end if;

     if Sig_Keep_vec(2) = "B" then
        text_io.put("I got B!");
        text_io.new_line;
     end if;

     if Sig_Keep_vec(3) = "C" then
        text_io.put("I got C!");
        text_io.new_line;
     end if;

     if Sig_Keep_vec(4) = "D" then
        text_io.put("I got D!");
        text_io.new_line;
     end if;


TAE.Tae_Wpt.Wpt_SetStringConstraints(Info.Panel_Id,"sig_list",taeint(10),S_Si
g_Keep_Vec);

     Dummy:=TAE.Tae_Wpt.Wpt_Pending;
  ------------------------------------------------------------------------


  exception

     when TAE.Tae_Wpt.BAD_STATE =>
       Text_IO.Put_Line ("Panel_sdetail.Connect_Panel:  "
         & "Invalid panel state.");
       raise;

  end Connect_Panel;


  --  ...................................................................
  --  .
  --  .    Destroy_Panel                    -- Subprogram BODY
  --  .
  --  ...................................................................
```

```
procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_sdetail.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;



--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--   .......................................................................
--   .
--   .    sdetail_cancel_Event              -- Subprogram SPEC & BODY
--   .
--   .......................................................................

procedure sdetail_cancel_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: sdetail_cancel.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: sdetail_cancel.
```

```
begin -- sdetail_cancel_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel sdetail, parm sdetail_cancel: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- sdetail_cancel
  Destroy_Panel;
  Panel_opdetail.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: sdetail_cancel.
  -- MERGE NOTE: Add code ABOVE this line for parm: sdetail_cancel.

end sdetail_cancel_Event;



--  ..................................................................
--  .
--  .    sdetail_help_Event               -- Subprogram SPEC & BODY
--  .
--  ..................................................................

procedure sdetail_help_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: sdetail_help.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: sdetail_help.

begin -- sdetail_help_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel sdetail, parm sdetail_help: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;
  -- sdetail_help
  Destroy_Panel;
  Panel_opdetail.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: sdetail_help.
  -- MERGE NOTE: Add code ABOVE this line for parm: sdetail_help.

end sdetail_help_Event;



--  .....................................................................
--  .
--  .    sdetail_next_Event              -- Subprogram SPEC & BODY
--  .
--  .....................................................................

procedure sdetail_next_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: sdetail_next.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: sdetail_next.

begin -- sdetail_next_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel sdetail, parm sdetail_next: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- sdetail_next
  Destroy_Panel;
  Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: sdetail_next.
  -- MERGE NOTE: Add code ABOVE this line for parm: sdetail_next.

end sdetail_next_Event;



--  .....................................................................
--   .
--   .    sdetail_ok_Event                  -- Subprogram SPEC & BODY
--   .
--   ..........................................................................

procedure sdetail_ok_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)

  Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
  Count : TAE.Taeint;

  -- MERGE NOTE: Add declarations BELOW this line for parm: sdetail_ok.
  -- MERGE NOTE: Add declarations ABOVE this line for parm: sdetail_ok.

begin -- sdetail_ok_Event

  TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
  Text_IO.Put ("Panel sdetail, parm sdetail_ok: value = ");
  if Count > 0 then
    TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
    Text_IO.Put_Line (Value(1));
  else
    Text_IO.Put_Line ("none");
  end if;

  -- sdetail_ok
  Destroy_Panel;
  Panel_opdetail.Connect_Panel (TAE.Tae_Wpt.WPT_PREFERRED);

  -- MERGE NOTE: Add code BELOW this line for parm: sdetail_ok.
  -- MERGE NOTE: Add code ABOVE this line for parm: sdetail_ok.

end sdetail_ok_Event;



--   ..........................................................................
--   .
--   .    sig_display_Event                 -- Subprogram SPEC & BODY
--   .
--   ..........................................................................

procedure sig_display_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)
```

```
    Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
    Count : TAE.Taeint;

    -- MERGE NOTE: Add declarations BELOW this line for parm: sig_display.
    -- MERGE NOTE: Add declarations ABOVE this line for parm: sig_display.

  begin -- sig_display_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel sdetail, parm sig_display: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: sig_display.
    -- MERGE NOTE: Add code ABOVE this line for parm: sig_display.

  end sig_display_Event;



  --  ..........................................................................
  --  .
  --  .    sig_list_Event                   -- Subprogram SPEC & BODY
  --  .
  --  ..........................................................................

  procedure sig_list_Event
    ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| PURPOSE:
  --| EVENT HANDLER.  Insert application specific information.
  --|
  --| NOTES: (none)

    Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
    Count : TAE.Taeint;

    -- MERGE NOTE: Add declarations BELOW this line for parm: sig_list.
    -- MERGE NOTE: Add declarations ABOVE this line for parm: sig_list.

 ---------------------------------------------------------------
 --ADDED By RUEY-WEN (VINCENT) HONG 960228
 ---------------------------------------------------------------
    S_Value : TAE.s_vector(1..1);

  begin -- sig_list_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel sdetail, parm sig_list: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: sig_list.
    -- MERGE NOTE: Add code ABOVE this line for parm: sig_list.

 -------------------------------------------------------------------
 --ADDED By RUEY-WEN (VINCENT) HONG 960228
 -------------------------------------------------------------------

  S_Value(1) := new String (1..TAE.Tae_Taeconf.STRINGSIZE);

  S_Value(1).all := Value(1);

  TAE.TAE_Vm.Vm_SetString_Vec(Info.target, "sig_display", 1, S_Value,
TAE.TAE_Vm.P_Update);
  TAE.TAE_Wpt.Wpt_ParmUpdate(Info.Panel_Id, "sig_display");
 -------------------------------------------------------------------

  end sig_list_Event;



  --  ......................................................................
  --  .
  --  .    sig_name_keyin_Event              -- Subprogram SPEC & BODY
  --  .
  --  ......................................................................

  procedure sig_name_keyin_Event
    ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| PURPOSE:
  --| EVENT HANDLER.  Insert application specific information.
  --|
  --| NOTES: (none)

    Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
    Count : TAE.Taeint;

    -- MERGE NOTE: Add declarations BELOW this line for parm: sig_name_keyin.
    -- MERGE NOTE: Add declarations ABOVE this line for parm: sig_name_keyin.

  begin -- sig_name_keyin_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel sdetail, parm sig_name_keyin: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: sig_name_keyin.
    -- MERGE NOTE: Add code ABOVE this line for parm: sig_name_keyin.

  end sig_name_keyin_Event;



  --
  -- end EVENT HANDLERS
  --
  ------------------------------------------------------------------------



  --  ....................................................................
  --  .
  --  .    Dispatch_Item                    -- Subprogram BODY
  --  .
  --  ....................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| NOTES: (none)

  begin -- Dispatch_Item

    if TAE.Tae_Misc.s_equal ("sdetail_cancel", User_Context_Ptr.Parm_Name)
then
      sdetail_cancel_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("sdetail_help", User_Context_Ptr.Parm_Name)
then
      sdetail_help_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("sdetail_next", User_Context_Ptr.Parm_Name)
then
      sdetail_next_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("sdetail_ok", User_Context_Ptr.Parm_Name) then
      sdetail_ok_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("sig_display", User_Context_Ptr.Parm_Name)
then
      sig_display_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("sig_list", User_Context_Ptr.Parm_Name) then
      sig_list_Event (User_Context_Ptr);
    elsif TAE.Tae_Misc.s_equal ("sig_name_keyin",
User_Context_Ptr.Parm_Name) then
      sig_name_keyin_Event (User_Context_Ptr);
    end if;

  end Dispatch_Item;
```

```
-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

end Panel_sdetail;


-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_select_1_s.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author     : Ruey-Wen (Vincent) Hong
-- *** Date       : Mar 5, 1996
-- *** Application: CAPS Software Base

--  *************************************************************************
--  *
--  *   Panel_select_1                      -- Package SPEC
--  *
--  *************************************************************************

with TAE;
with X_Windows;

package Panel_select_1 is

--| PURPOSE:
--| This package encapsulates the TAE Plus panel:  select_1
--| These subprograms enable panel initialization, creation, destruction,
--| and event dispatching.  For more advanced manipulation of the panel
--| using the TAE package, the panel's Event_Context (Info) is provided.
--| It includes the Target and View (available after initialization)
--| and the Panel_Id (available after creation).
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| The following Workbench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  select_1
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated


  Info : TAE.Tae_Wpt.Event_Context_Ptr;    -- panel information


  --  ......................................................................
  --  .
  --  .   Initialize_Panel                   -- Subprogram SPEC
  --  .
  --  ......................................................................
```

```
procedure Initialize_Panel
  ( Collection_Read                          -- TAE Collection read from
      : in TAE.Tae_Co.Collection_Ptr );     -- resource file

--| PURPOSE:
--| This procedure initializes the Info.Target and Info.View for this panel
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if Collection_Read not initialized
--| TAE.Tae_Co.NO_SUCH_MEMBER is raised if the panel is not in
--|    Collection_Read
--|
--| NOTES: (none)




--  ......................................................................
--  .
--  .    Create_Panel                        -- Subprogram SPEC
--  .
--  ......................................................................

procedure Create_Panel
  ( Panel_State                              -- Flags sent to Wpt_NewPanel.
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                          -- Panel origin is offset from
      : in X_Windows.Window                  -- this X Window.  Null_Window
        := X_Windows.Null_Window );          -- uses the root window.

--| PURPOSE:
--| This procedure creates this panel object in the specified Panel_State
--| and stores the panel Id in Info.Panel_Id.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised if the panel is not initialized
--| TAE.TAE_FAIL is raised if the panel could not be created
--|
--| NOTES: (none)



--  ......................................................................
--  .
--  .    Connect_Panel                       -- Subprogram SPEC
--  .
--  ......................................................................

procedure Connect_Panel
  ( Panel_State
        : in TAE.Tae_Wpt.Wpt_Flags
          := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window                        -- Panel origin is offset from
        : in X_Windows.Window              -- this X Window.  Null_Window
          := X_Windows.Null_Window );      -- uses the root window.

--| PURPOSE:
--| If this panel doesn't exist, this procedure creates this panel object
--|   in the specifiec Panel_State and stores the panel Id in
--|   Info.Panel_Id.
--| If this panel does exist, it is set to the specified Panel_State.
--|   In this case, Relative_Window is ignored.
--|
--| EXCEPTIONS:
--| TAE.UNINITIALIZED_PTR is raised from Create_Panel if the panel is
--|    not initialized
--| TAE.TAE_FAIL is raised from Create_Panel if the panel could not be
--|    created
--| TAE.Tae_Wpt.BAD_STATE is raised if the panel exists and the
--|    Panel_State is an invalid state
--|
--| NOTES: (none)



--   .....................................................................
--   .
--   .    Destroy_Panel                    -- Subprogram SPEC
--   .
--   .....................................................................

procedure Destroy_Panel;

--| PURPOSE:
--| This procedure erases a panel from the screen and de-allocates the
--| associated panel object (not the target and view).
--|
--| EXCEPTIONS:
--| TAE.Tae_Wpt.BAD_PANEL_ID is raised if Info.Panel_Id is an invalid id.
--|
--| NOTES:
--| Info.Panel_Id is set to TAE.NULL_PANEL_ID, and should not referenced
--| in any Wpt call until it is created again.




--   .....................................................................
--   .
--   .    Dispatch_Item                    -- Subprogram SPEC
--   .
--   .....................................................................
```

```
  procedure Dispatch_Item
    ( User_Context_Ptr                      -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr ); -- event.

  --| PURPOSE:
  --| This procedure calls the Event Handler specified by User_Context_Ptr
  --|
  --| EXCEPTIONS:
  --| Application-specific
  --|
  --| NOTES: (none)

end Panel_select_1;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:        pan_select_1_b.a
-- *** Generated:   Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--   ***************************************************************************
--   *
--   *    Panel_select_1                    -- Package BODY
--   *
--   ***************************************************************************
with TAE; use TAE;
with Text_IO;
with Global;

-- One "with" statement for each connected panel.

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body Panel_select_1 is

--| NOTES:
--| For each parameter that you have defined to be "event-generating" in
--| this panel, there is an event handler procedure below.  Each handler
--| has a name that is a concatenation of the parameter name and "_Event".
--| Add application-dependent logic to each event handler.  (As generated
--| by the WorkBench, each event handler simply logs the occurrence of
--| the event.)
--|
--| For best automatic code merging results, you should put as many
--| modifications as possible between the lines of the MERGE NOTE comments.
--| Modifications outside the MERGE NOTEs will often merge correctly, but
--| must sometimes be merged by hand.  If the modifications cannot be
--| automatically merged, a reject file (*.rej) will be generated which
--| will contain your modifications.
--|
--| REGENERATED:
```

```
--| The following WorkBench operations will cause regeneration of this file:
--|     The panel's name is changed (not title)
--| For panel:  select_1
--|
--| The following WorkBench operations will also cause regeneration:
--|     An item is deleted
--|     A new item is added to this panel
--|     An item's name is changed (not title)
--|     An item's data type is changed
--|     An item's generates events flag is changed
--|     An item's valids changed (if item is type string and connected)
--|     An item's connection information changed
--| For the panel items:
--|   select_1_disp
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

  --  ......................................................................
  --  .
  --  .    Initialize_Panel                  -- Subprogram BODY
  --  .
  --  ......................................................................

  procedure Initialize_Panel
    ( Collection_Read
        : in TAE.Tae_Co.Collection_Ptr ) is

  --| NOTES: (none)

    -- MERGE NOTE: Add declarations for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add declarations for Initialize_Panel ABOVE this line.

  begin -- Initialize_Panel

    Info := new TAE.Tae_Wpt.Event_Context;
    Info.Collection := Collection_Read;
    TAE.Tae_Co.Co_Find (Info.Collection, "select_1_v", Info.View);
    TAE.Tae_Co.Co_Find (Info.Collection, "select_1_t", Info.Target);

    -- MERGE NOTE: Add code for Initialize_Panel BELOW this line.
    -- MERGE NOTE: Add code for Initialize_Panel ABOVE this line.

  exception

    when TAE.UNINITIALIZED_PTR =>
      Text_IO.Put_Line ("Panel_select_1.Initialize_Panel:  "
        & "Collection_Read not initialized.");
    raise;

  when TAE.Tae_Co.NO_SUCH_MEMBER =>
    Text_IO.Put_Line ("Panel_select_1.Initialize_Panel:  "
      & "(View or Target) not in Collection.");
    raise;

end Initialize_Panel;



--  ...................................................................
--  .
--  .    Create_Panel                          -- Subprogram BODY
--  .
--  ...................................................................

procedure Create_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Create_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Create_Panel ABOVE this line.

begin -- Create_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    TAE.Tae_Wpt.Wpt_NewPanel
      ( Display_Id             => Global.Default_Display_Id,
        Data_Vm                => Info.Target,
        View_Vm                => Info.View,
        Relative_Window        => Relative_Window,
        User_Context           => Info,
        Flags                  => Panel_State,
        Panel_Id               => Info.Panel_Id );
  else
    Text_IO.Put_Line ("Panel (select_1) is already displayed.");
  end if;

  -- MERGE NOTE: Add code for Create_Panel BELOW this line.
  -- MERGE NOTE: Add code for Create_Panel ABOVE this line.

exception

  when TAE.UNINITIALIZED_PTR =>
    Text_IO.Put_Line ("Panel_select_1.Create_Panel:  "
      & "Panel was not initialized prior to creation.");
    raise;

  when TAE.TAE_FAIL =>
    Text_IO.Put_Line ("Panel_select_1.Create_Panel:  "
      & "Panel could not be created.");
    raise;

end Create_Panel;



--  .....................................................................
--  .
--  .    Connect_Panel                    -- Subprogram BODY
--  .
--  .....................................................................

procedure Connect_Panel
  ( Panel_State
      : in TAE.Tae_Wpt.Wpt_Flags
        := TAE.Tae_Wpt.WPT_PREFERRED;

    Relative_Window
      : in X_Windows.Window
        := X_Windows.Null_Window ) is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Connect_Panel ABOVE this line.

begin -- Connect_Panel

  if Info.Panel_Id = Tae.Null_Panel_Id then
    Create_Panel
      ( Relative_Window       => Relative_Window,
        Panel_State           => Panel_State );
  else
    TAE.Tae_Wpt.Wpt_SetPanelState (Info.Panel_Id, Panel_State);
  end if;

  -- MERGE NOTE: Add code for Connect_Panel BELOW this line.
  -- MERGE NOTE: Add code for Connect_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_STATE =>
    Text_IO.Put_Line ("Panel_select_1.Connect_Panel:  "
      & "Invalid panel state.");
    raise;

end Connect_Panel;
```

```
--  ..........................................................................
--  .
--  .    Destroy_Panel                        -- Subprogram BODY
--  .
--  ..........................................................................

procedure Destroy_Panel is

--| NOTES: (none)

  -- MERGE NOTE: Add declarations for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add declarations for Destroy_Panel ABOVE this line.

begin -- Destroy_Panel

  TAE.Tae_Wpt.Wpt_PanelErase(Info.Panel_Id);

  -- MERGE NOTE: Add code for Destroy_Panel BELOW this line.
  -- MERGE NOTE: Add code for Destroy_Panel ABOVE this line.

exception

  when TAE.Tae_Wpt.BAD_PANEL_ID =>
    Text_IO.Put_Line ("Panel_select_1.Destroy_Panel:  "
      & "Info.Panel_Id is an invalid id.");
    raise;

  when TAE.Tae_Wpt.ERASE_NULL_PANEL =>
    -- This panel has not been created yet, or has already been destroyed.
    -- Trap this exception and do nothing.
    null;

end Destroy_Panel;



--+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
--
-- begin EVENT HANDLERS
--
--  ..........................................................................
--  .
--  .    select_1_disp_Event                  -- Subprogram SPEC & BODY
--  .
--  ..........................................................................

procedure select_1_disp_Event
  ( Info : in TAE.Tae_Wpt.Event_Context_Ptr ) is

--| PURPOSE:
--| EVENT HANDLER.  Insert application specific information.
--|
--| NOTES: (none)
```

```
    Value : array (1..1) of String (1..TAE.Tae_Taeconf.STRINGSIZE);
    Count : TAE.Taeint;

    -- MERGE NOTE: Add declarations BELOW this line for parm: select_1_disp.
    -- MERGE NOTE: Add declarations ABOVE this line for parm: select_1_disp.

  begin -- select_1_disp_Event

    TAE.Tae_Vm.Vm_Extract_Count (Info.Parm_Ptr, Count);
    Text_IO.Put ("Panel select_1, parm select_1_disp: value = ");
    if Count > 0 then
      TAE.Tae_Vm.Vm_Extract_SVAL (Info.Parm_Ptr, 1, Value(1));
      Text_IO.Put_Line (Value(1));
    else
      Text_IO.Put_Line ("none");
    end if;

    -- MERGE NOTE: Add code BELOW this line for parm: select_1_disp.
    -- MERGE NOTE: Add code ABOVE this line for parm: select_1_disp.

  end select_1_disp_Event;



  --
  -- end EVENT HANDLERS
  --
  -------------------------------------------------------------------------



  --  ....................................................................
  --  .
  --  .    Dispatch_Item                    -- Subprogram BODY
  --  .
  --  ....................................................................

  procedure Dispatch_Item
    ( User_Context_Ptr : in TAE.Tae_Wpt.Event_Context_Ptr ) is

  --| NOTES: (none)

  begin -- Dispatch_Item

    if TAE.Tae_Misc.s_equal ("select_1_disp", User_Context_Ptr.Parm_Name) then
      select_1_disp_Event (User_Context_Ptr);
    end if;

  end Dispatch_Item;


-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.
```

```
end Panel_select_1;


-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         sbgui.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author       : Ruey-Wen (Vincent) Hong
-- *** Date         : Mar 5, 1996
-- *** Application: CAPS Software Base

--   ...........................................................................
--   .
--   .    sbgui                               -- Main program BODY
--   .
--   ...........................................................................

--| PURPOSE:
--| This is the main program of an application generated by the TAE Plus
--| Code Generator.
--|
--| EXCEPTIONS: (none)
--|
--| REGENERATED:
--| This file is generated only once.  Therefore, you may modify it without
--| impacting automatic code merge.
--|
--| NOTES:
--| To turn this into a real application, do the following:
--|
--| 1.  Each panel that has event generating parameters is encapsulated by
--| a package.  Each parameter that you have defined to be "event-generating",
--| has an event handler procedure in the appropriate panel's package body.
--| The panel's package body is in a file, named by concatenating the string
--| "pan_" with the panel name (followed by "_b.a").  Each handler has a name
--| that is a concatenation of the parameter name and the string "_Event"
--| Add application-dependent logic to each event handler.
--| (As generated by the Workbench, each event handler simply logs the
--| occurrence of the event.)
--|
--| 2.  See the "TAE Plus Ada Programmer's Guide" for directions on how
--| to compile and link this program.
--|
--|
--| ADDITIONAL NOTES:
--| In TAE Plus version 4.1, every TAE Ada application caused the screen
--| to be cleared as part of the initialization of the tae_termio package.
--| In TAE Plus version 5.0, the TAE Ada bindings have been changed so as
--| not to clear the screen by default.  Simply adding a call to
--| TAE.Tae_Termio.T_Clear will produce the same results as 4.1.
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated
--|-----------------------------------------------------------------------------
```

```
with TAE;
with Text_IO;
with Global;                                    -- application global variables

-- PROGRAMMER NOTE:
-- add one "with" for each resource file in this application
with sbgui_Support;


procedure sbgui is

  Wpt_Event                : TAE.Tae_Wpt.Wpt_Eventptr;
  Type_of_Wpt_Event        : Tae.Wpt_Eventtype;
  User_Context_Ptr         : TAE.Tae_Wpt.Event_Context_Ptr;
  Panel_In_Resource_File : Boolean;
  UNKNOWN_WPT_EVENT        : Exception;

begin -- sbgui

  -- permit upper/lowercase file names
  TAE.Tae_Misc.F_Force_Lower (FALSE);

  TAE.Tae_Wpt.Wpt_Init ("", Global.Default_Display_Id);
  -- PROGRAMMER NOTE:
  -- To enable scripting, uncomment the following line.  See the
  -- taerecord man page.
  --     TAE.Tae_Wpt.Wpt_ScriptInit ("sbgui");

  -- initialize resource file
  -- PROGRAMMER NOTE:
  -- For each resource file in this application, calls to the appropriate
  -- Initialize_All_Panels and Create_Initial_Panels must be added.
  sbgui_Support.Initialize_All_Panels("/tmp_mnt/users/work2/vincent/caps95/
src/combine.work2/sbgui.res");
  sbgui_Support.Create_Initial_Panels;

  TAE.Tae_Wpt.Wpt_NewEvent (Wpt_Event);


EVENT_LOOP:
  while not Global.Application_Done loop

    -- PROGRAMMER NOTE:
    -- use Global.Set_Application_Done in "quit" event handler to exit loop

    -- Wait for the next event
    --
    TAE.Tae_Wpt.Wpt_NextEvent
      ( Event => Wpt_Event,
        Etype => Type_of_Wpt_Event );

    case Type_of_Wpt_Event is
```

```
when TAE.Tae_Wpt.WPT_PARM_EVENT =>

  -- Panel event has occurred.

  -- PROGRAMMER NOTE:
  -- You can comment out the the following "put" call.
  -- The appropriate EVENT_HANDLER finishes the message.
  --
  Text_IO.Put ("Event: WPT_PARM_EVENT, ");

  -- Get the user context (which is stored in the panel object
  -- when Wpt_NewPanel is called).
  --
  TAE.Tae_Wpt.Wpt_Extract_Context
    ( Event    => Wpt_Event,
      User_Ptr => User_Context_Ptr );

  -- Get the parameter name
  --
  TAE.Tae_Wpt.Wpt_Extract_Parm
    ( Event    => Wpt_Event,
      Parm     => User_Context_Ptr.Parm_Name );

  -- Get target id
  --
  TAE.Tae_Wpt.Wpt_Extract_Data
    ( Event    => Wpt_Event,
      Data     => User_Context_Ptr.Datavm_Ptr );

  -- Find Vm parm object
  --
  TAE.Tae_Vm.Vm_Find
    ( Vmid     => User_Context_Ptr.Datavm_Ptr,
      Name     => User_Context_Ptr.Parm_Name,
      Vout     => User_Context_Ptr.Parm_Ptr );

  -- Dispatch event to event handler
  --
  sbgui_Support.Dispatch_Panel
    ( User_Context_Ptr       => User_Context_Ptr,
      Panel_In_Resource_File => Panel_In_Resource_File );

  if (not Panel_In_Resource_File) then

    -- PROGRAMMER NOTE:
    -- For applications with more than one resource file,
    -- add a call to Dispatch_Panel for each resource file.
    --
    Text_IO.Put_Line ("Unexpected event from wpt!");
    raise TAE.Tae_Wpt.BAD_EVENT_ID;

  end if;
```

```
      when TAE.Tae_Wpt.WPT_FILE_EVENT =>

        Text_IO.Put_Line ("No EVENT_HANDLER for event from external source.");

        -- PROGRAMMER NOTE:
        -- Add code here to handle file events.
        -- Use Wpt_AddEvent and Wpt_RemoveEvent to register and remove
        -- event sources.
        -- Use Wpt_Extract_EventSource and Wpt_Extract_EventMask to get
        -- information about the event that occurred.


      when TAE.Tae_Wpt.WPT_WINDOW_EVENT =>

        null;

        -- PROGRAMMER NOTE:
        -- Add code here to handle window events.
        -- WPT_WINDOW_EVENT can be caused by windows which you directly create
        -- with X (not TAE panels), or by user acknowledgement of a
        -- Wpt_PanelMessage (therefore no default put_line statement is
        -- generated here).
        -- You MIGHT want to use Wpt_Extract_xEvent_Type here.
        --
        -- DO NOT use Wpt_Extract_Parm_xEvent since this is not
        -- a WPT_PARM_EVENT; you may get a "storage error".


      when TAE.Tae_Wpt.WPT_TIMEOUT_EVENT =>

        Text_IO.Put_Line ("STUB: Event WPT_TIMEOUT_EVENT");

        -- PROGRAMMER NOTE:
        -- Add code here to handle timeout events.
        -- Timeout events occur when an application has not received any
        -- user input within the interval specified by Wpt_SetTimeOut.


      when TAE.Tae_Wpt.WPT_TIMER_EVENT =>

        Text_IO.Put_Line ("No EVENT_HANDLER for timer.");

        -- PROGRAMMER NOTE:
        -- Add code here to handle timer events.
        -- Timer events occur on (or after) the interval specified when the
        -- event is registered using Wpt_AddTimer.  Use Wpt_RemoveTimer to
        -- remove timers.
        -- Use Wpt_Extract_TimerId, Wpt_Extract_TimerRepeat, and
        -- Wpt_Extract_TimerInterval to get information about the event
        -- that occurred.


      -- These are internal TAE events.  The application should never see
them.
```

```
      -- when TAE.Tae_Wpt.WPT_HELP_EVENT =>
      -- when TAE.Tae_Wpt.WPT_INTERRUPT_EVENT =>

      when others =>

        raise UNKNOWN_WPT_EVENT;

    end case;

  end loop EVENT_LOOP;
  TAE.Tae_Wpt.Wpt_Finish;   -- close all display connections

  -- PROGRAMMER NOTE:
  -- Application has ended normally.  Add application specific code to
  -- close down your application.

exception

  when UNKNOWN_WPT_EVENT =>
    Text_IO.Put ("FATAL ERROR: Unknown Wpt_NextEvent Event Type: ");
    Text_IO.Put (TAE.Wpt_Eventtype'image(Type_of_Wpt_Event) ) ;
    Text_IO.Put_Line (" ... Forced exit.");

end sbgui;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         sbgui__creat_init.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--   ..................................................................
--   .
--   .    Create_InitialPanels               -- Subprogram SUBUNIT
--   .
--   ..................................................................

separate (sbgui_Support)
procedure Create_Initial_Panels is

--| NOTES:
--| This subprogram is not in the same file as sbgui_Support.a
--| for code regeneration purposes.  Therefore it is a subunit.
--| Also note, that the parent unit "with"ed in all of the panel packages
--| for the entire resource file, so this unit doesn't need to "with" in
--| any of the panel packages.
--|
--| This procedure should be called after the panels in the initial panel
--| set have been initialized.
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     A panel is added to the initial panel set
--|     A panel is deleted from the initial panel set
--| For the set of initial panels:
--|   sb_main
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.

-- MERGE NOTE: Add additional code BELOW this line.
-- MERGE NOTE: Add additional code ABOVE this line.

begin -- Create_Initial_Panels

  Panel_sb_main.Create_Panel;

  -- MERGE NOTE: Add additional code BELOW this line.
  -- MERGE NOTE: Add additional code ABOVE this line.

end Create_Initial_Panels;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         sbgui_support_s.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--   *************************************************************************
--   *
--   *    sbgui_Support                         -- Package SPEC
--   *
--   *************************************************************************

with TAE;

package sbgui_Support is

--| PURPOSE:
--| This package encapsulates the operations specific to a resource file.
--| These subprograms enable initialization of all panels, displaying
--| all initial panels, and event dispatching.
--|
--| INITIALIZATION EXCEPTIONS: (none)
--|
--| NOTES: (none)
--|
--| REGENERATED:
--| This file is generated only once.  Therefore, you may modify it without
--| impacting automatic code merge.
--|
--| CHANGE LOG:
--|  4-Mar-96   TAE     Generated
```

```
--  .............................................................
--  .
--  .    Initialize_All_Panels           -- Subprogram SPEC
--  .
--  .............................................................

procedure Initialize_All_Panels
  ( Resource_File                        -- Name of resource file.
      : in String );

--| PURPOSE:
--| This procedure initializes all the panels in the resource file.
--|
--| EXCEPTIONS:
--| TAE.TAE_FAIL is raised when the resource file could not be read
--| TAE.Tae_Co.NO_SUCH_MEMBER is raised when one of the panels was not in
--|    the resource file.  This could happen if a panel was deleted from
--|    the resource file (using the WorkBench) after the code was
--|    generated.
--|
--| NOTES:
--| This procedure reads in the resource file and initializes each panel.
--| It only needs to be called once.



--  .............................................................
--  .
--  .    Create_Initial_Panels           -- Subprogram SPEC
--  .
--  .............................................................

procedure Create_Initial_Panels;

--| PURPOSE:
--| This procedure displays the set of initial panels.
--|
--| PARAMETERS: (none)
--|
--| EXCEPTIONS: (none)
--|
--| NOTES: (none)


--  .............................................................
--  .
--  .    Dispatch_Panel                  -- Subprogram SPEC
--  .
--  .............................................................

procedure Dispatch_Panel
  ( User_Context_Ptr                     -- Wpt Event Context for a PARM
        : in TAE.Tae_Wpt.Event_Context_Ptr; -- event.

      Panel_In_Resource_File              -- True, if the PARM event
        : out Boolean );                  -- occurred in one of the panels
                                          -- of this resource file.
                                          -- False, otherwise.
   --| PURPOSE:
   --| This procedure dispatches a WPT_PARM_EVENT to the appropriate panel's
   --| Dispatch_Item routine.
   --|
   --| EXCEPTIONS:
   --| Application-specific
   --|
   --| NOTES: (none)

end sbgui_Support;

-- *** TAE Plus Code Generator version V5.3 [Merge Token: DO NOT DELETE.]
-- *** File:         sbgui_support_b.a
-- *** Generated:    Mar  4 14:14:09 1996
-- *** Author      : Ruey-Wen (Vincent) Hong
-- *** Date        : Mar 5, 1996
-- *** Application: CAPS Software Base

--   **************************************************************************
--   *
--   *    sbgui_Support                       -- Package BODY
--   *
--   **************************************************************************

with TAE;
with Text_IO;

-- one "with" for each panel in the resource file
with Panel_backup_l;
with Panel_create_l;
with Panel_delete_l;
with Panel_go;
with Panel_grnd_eq;
with Panel_keyword;
with Panel_maintain;
with Panel_mbn_kps;
with Panel_op_no;
with Panel_opdetail;
with Panel_query;
with Panel_sb_main;
with Panel_sdetail;
with Panel_select_l;

-- MERGE NOTE: Add additional "with"s BELOW this line.
-- MERGE NOTE: Add additional "with"s ABOVE this line.

package body sbgui_Support is
```

```
--| NOTES: (none)
--|
--| REGENERATED:
--| The following WorkBench operations will cause regeneration of this file:
--|     A panel is deleted
--|     A new panel is added
--|     A panel's name is changed (not title)
--| For the panels:
--|   backup_1, create_1, delete_1, go,       grnd_eq,  keyword,  maintain,
--|   mbn_kps,  op_no,    opdetail, query,    sb_main,  sdetail,  select_1,
--|
--| CHANGE LOG:
--| MERGE NOTE: Add Change Log entries BELOW this line.
--|  4-Mar-96   TAE     Generated
--| MERGE NOTE: Add Change Log entries ABOVE this line.


  --  .....................................................................
  --  .
  --  .    Create_Initial_Panels            -- Subprogram STUB
  --  .
  --  .....................................................................

  procedure Create_Initial_Panels is separate;




  --  .....................................................................
  --  .
  --  .    Initialize_All_Panels            -- Subprogram BODY
  --  .
  --  .....................................................................

  procedure Initialize_All_Panels
    ( Resource_File
        : in  String ) is

  --| NOTES: (none)

    Vm_Collection_Read   : TAE.Tae_Co.Collection_Ptr;

    -- MERGE NOTE: Add additional declarations BELOW this line.
    -- MERGE NOTE: Add additional declarations ABOVE this line.

  begin -- Initialize_All_Panels

  -- MERGE NOTE: Add additional code BELOW this line. (1)
  -- MERGE NOTE: Add additional code ABOVE this line. (1)

    -- do one Co_New and Co_ReadFile per resource file
    TAE.Tae_Co.Co_New
      ( Flags => 0,
        Coid   => Vm_Collection_Read );

    -- could pass P_Abort if you prefer
    TAE.Tae_Co.Co_ReadFile
      ( Coid => Vm_Collection_Read,
        Spec => Resource_File,
        Mode => TAE.P_CONT );

    Panel_backup_1.Initialize_Panel (Vm_Collection_Read);
    Panel_create_1.Initialize_Panel (Vm_Collection_Read);
    Panel_delete_1.Initialize_Panel (Vm_Collection_Read);
    Panel_go.Initialize_Panel (Vm_Collection_Read);
    Panel_grnd_eq.Initialize_Panel (Vm_Collection_Read);
    Panel_keyword.Initialize_Panel (Vm_Collection_Read);
    Panel_maintain.Initialize_Panel (Vm_Collection_Read);
    Panel_mbn_kps.Initialize_Panel (Vm_Collection_Read);
    Panel_op_no.Initialize_Panel (Vm_Collection_Read);
    Panel_opdetail.Initialize_Panel (Vm_Collection_Read);
    Panel_query.Initialize_Panel (Vm_Collection_Read);
    Panel_sb_main.Initialize_Panel (Vm_Collection_Read);
    Panel_sdetail.Initialize_Panel (Vm_Collection_Read);
    Panel_select_1.Initialize_Panel (Vm_Collection_Read);

    -- MERGE NOTE: Add additional code BELOW this line.
    -- MERGE NOTE: Add additional code ABOVE this line.

  exception

    when TAE.TAE_FAIL =>
      Text_IO.Put_Line ("resfile_Support.Initialize_All_Panels:  ");
      Text_IO.Put_Line (Resource_File
        & " doesn't exist or is incorrectly formatted.");
      raise;

    when TAE.Tae_Co.NO_SUCH_MEMBER =>
      -- raised from one of the panel's Initialize_Panel
      raise;

  end Initialize_All_Panels;



  --  ......................................................................
  --  .
  --  .    Dispatch_Panel                   -- Subprogram BODY
  --  .
  --  ......................................................................

  procedure Dispatch_Panel
    ( User_Context_Ptr
        : in  TAE.Tae_Wpt.Event_Context_Ptr;

      Panel_In_Resource_File
```

```
          : out Boolean ) is

   --| NOTES: (none)

   begin -- Dispatch_Parm_Event

     Panel_In_Resource_File := TRUE;
     if TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_backup_1.Info) then
       Panel_backup_1.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_create_1.Info) then
       Panel_create_1.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_delete_1.Info) then
       Panel_delete_1.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_go.Info) then
       Panel_go.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_grnd_eq.Info) then
       Panel_grnd_eq.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_keyword.Info) then
       Panel_keyword.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_maintain.Info) then
       Panel_maintain.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_mbn_kps.Info) then
       Panel_mbn_kps.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_op_no.Info) then
       Panel_op_no.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_opdetail.Info) then
       Panel_opdetail.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_query.Info) then
       Panel_query.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_sb_main.Info) then
       Panel_sb_main.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_sdetail.Info) then
       Panel_sdetail.Dispatch_Item (User_Context_Ptr);
     elsif TAE.Tae_Wpt."=" (User_Context_Ptr, Panel_select_1.Info) then
       Panel_select_1.Dispatch_Item (User_Context_Ptr);
     else
       Panel_In_Resource_File := FALSE;
     end if;

   end Dispatch_Panel;


   -- MERGE NOTE: Add additional code BELOW this line. (2)
   -- MERGE NOTE: Add additional code ABOVE this line. (2)

 end sbgui_Support;
```

```
# *** TAE Plus Code Generator version V5.3 ***
# *** File:             Make.sbgui ***
# *** Generated:        Feb 22 12:14:44 1996 ***
# *** Author:           Ruey-Wen (Vincent) Hong
# *** Date:             Mar 5, 1996
# *** Application:      CAPS Software Base
#####################################################################
# PURPOSE:
# This file is included into the Makefile.  It lists the generated
# source files that are specific to the resource file:
#  /tmp_mnt/users/work2/vincent/caps95/src/combine.work/sbgui.res
#
# REGENERATED:
# The following WorkBench operations will cause regeneration of this file:
#     A panel is deleted
#     A new panel is added
#     A panel's name is changed (not title)
# For the panels:
#   backup_1, create_1, delete_1, go,       grnd_eq,  keyword,  maintain,
#   mbn_kps,  op_no,    opdetail, query,    sb_main,  sdetail,  select_1,
#
#####################################################################
SRCS_sbgui = \
        pan_backup_1_s.a \
        pan_backup_1_b.a \
        pan_create_1_s.a \
        pan_create_1_b.a \
        pan_delete_1_s.a \
        pan_delete_1_b.a \
        pan_go_s.a \
        pan_go_b.a \
        pan_grnd_eq_s.a \
        pan_grnd_eq_b.a \
        pan_keyword_s.a \
        pan_keyword_b.a \
        pan_maintain_s.a \
        pan_maintain_b.a \
        pan_mbn_kps_s.a \
        pan_mbn_kps_b.a \
        pan_op_no_s.a \
        pan_op_no_b.a \
        pan_opdetail_s.a \
        pan_opdetail_b.a \
        pan_query_s.a \
        pan_query_b.a \
        pan_sb_main_s.a \
        pan_sb_main_b.a \
        pan_sdetail_s.a \
        pan_sdetail_b.a \
        pan_select_1_s.a \
        pan_select_1_b.a \
        sbgui_support_s.a \
        sbgui_support_b.a \
        sbgui__creat_init.a \
        gldef.a \
        asable.a \
        semops.a \
        utilop.a \
        utilso.a \
        nsa.a \
        sigmat.a \
        semat.a \
        fresult.a \
        swb_def.a \
        ring.a \
        swb.a \
        init.a



# *** TAE Plus Code Generator version V5.3 ***
# *** File:           Makefile ***
# *** Generated:      Feb 22 12:14:44 1996 ***
# *** Author:         Ruey-Wen (Vincent) Hong
# *** Date:           Mar 5, 1996
# *** Application:    CAPS Software Base
####################################################################
# PURPOSE:
# This is the Makefile of an Ada application generated by the TAE Plus
# Code Generator.
#
# REGENERATED:
# This file is generated only once.  Therefore, you may modify it without
# impacting automatic code merge.
#
# NOTES:
# 1.  This Makefile is just a wrapper around the Verdix supplied
# "a.make" and "a.mklib" commands.  If you are porting this code to
# another compiler, you will not be able to use this Makefile.
#
# 2.  To build your application, type "make".
####################################################################

        PROGRAM = sbgui

         ADALIB = .
    AMAKE_FLAGS = -v -L $(ADALIB)
   AMKLIB_FLAGS = -v
        LDFLAGS = -o $(PROGRAM)

# PROGRAMMER NOTE:
# Add a line '#include Make.RESFILENAME' for each resource file in
# your application.
#
include Make.sbgui

# PROGRAMMER NOTE:
```

```
# Add $(SRCS_RESFILENAME) for each resource file in your application.
#
        GENSRCS = $(PROGRAM).a global_s.a global_b.a \
                  $(SRCS_sbgui)

# PROGRAMMER NOTE:
# Add your application specific srcs (that are not generated by the
# code generator) here.
#
        APPSRCS =

           SRCS = $(APPSRCS) $(GENSRCS)


all:: mklibif $(PROGRAM)


$(PROGRAM)::
            a.make $(AMAKE_FLAGS) $(PROGRAM) $(LDFLAGS) -f $(SRCS)


# Only make the ada library if the file "ada.lib" doesn't exist.
#
mklibif::
            -@if [ ! -f ada.lib  ]; then \
                        echo "a.mklib $(AMKLIB_OPTS) $(ADALIB)
$(TAEADALIB)"; \
                        a.mklib $(AMKLIB_OPTS) $(ADALIB) $(TAEADALIB); \
            fi;


mklib::
            a.mklib $(AMKLIB_FLAGS) $(ADALIB) $(TAEADALIB)
```

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center 8725 John J. Kingman Road, Suite 0944 Fort Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Rd. Monterey, California 93943-5101	2
3.	Dr. Ted Lewis, Chairman, Code CS/Lt Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	1
4.	Chief of Naval Research 800 N. Quincy Street Arlington, VA 22217	1
5.	Dr. Luqi, Code CS/Lq Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	10
6.	Dr. Man-Tak Shing, Code CS/Sh Department of Computer Science Naval Postgraduate School Monterey, California 93943-5100	10
7.	Mr. Ruey-Wen Hong 26, Ave 22, Lane 119, Jui-Lin Road Yong-Ho, Taipei , Taiwan, R.O.C.	3
8.	Ada Joint Program Office OUSDRE (R&AT) The Pentagon Washington, DC 20301	1